Visibly Pushdown Automata with Multiplicities: Finiteness and K-Boundedness

Mathieu Caralp, Pierre-Alain Reynier, Jean-Marc Talbot

Visibly Pushdown Automata with Multiplicities: Finiteness and K-Boundedness

Mathieu Caralp, Pierre-Alain Reynier, and Jean-Marc Talbot

Laboratoire d'Informatique Fondamentale de Marseille, AMU & CNRS, UMR 7279

Abstract. We propose an extension of visibly pushdown automata by means of weights (represented as positive integers) associated with transitions, called visibly pushdown automata with multiplicities. The multiplicity of a computation is the product of the multiplicities of the transitions used along this computation. The multiplicity of an input is the sum of the ones of all its successful computations. Finally, the multiplicity of such an automaton is the supremum of multiplicities over all possible inputs.
We prove the problem of deciding whether the multiplicity of an automaton is finite to be in PTIME. We also consider the K-boundedness problem, *i.e.* deciding whether the multiplicity is bounded by K: we prove this problem to be EXPTIME-complete when K is part of the input and in PTIME when K is fixed.
As visibly pushdown automata are closely related to tree automata, we discuss deeply the relationship of our extension with weighted tree automata.

1 Introduction

Visibly pushdown automata (VPA for short) have been proposed in [1] as an interesting subclass of pushdown automata, strictly more expressive that finite state automata, but still enjoying good closure and decidability properties. They are pushdown automata such that the behavior of the stack, *i.e.* whether it pushes or pops, is visible in the input word. Technically, the input alphabet is partitioned into call, return and internal symbols. When reading a call the automaton must push a symbol onto the stack, when reading a return it must pop and when reading an internal it cannot touch the stack. The partitioning of the alphabet induces a nesting structure of the input word. Calls and returns can be viewed as opening/closing brackets, and well-nested words are words where every call symbol (resp. return symbol) has a matching return (resp. call).

The original motivation for their introduction was for verification purposes, the stack being used for the modelization of call/returns of functions. Another application domain is the processing of XML documents. Indeed, unranked trees in their linear form can be viewed as well-nested words. Actually, the model of visibly pushdown automata is expressively equivalent to that of finite tree automata, see [1].

It is quite standard to extend a class of automata with weights, by adding a labeling function assigning a weight to each transition. In this work, we consider VPA with multiplicities ($\mathbb{N}$-VPA for short) where weights are positive integers (multiplicities). The multiplicity of a run is the product of the multiplicities of the transitions used along it. The multiplicity of a word is the sum of the ones of all its accepting runs. Finally, the multiplicity of the automaton is the supremum of the multiplicities of the words it accepts. This model extends the model of finite state automata with multiplicities [10].

A special case of multiplicity is the degree of ambiguity of a word, *i.e.* the number of accepting runs (obtained when every transition has weight 1). The class of finitely ambiguous automata has been investigated for both automata on words and on trees [5,15,12,13]. The interest in this class arised from the fact that it allows an efficient (polynomial) equivalence check. An analogy can be drawn with the context of transducers where the equivalence problem is decidable for finite-valued transducers (and undecidable in general). In [11], the characterization of automata whose multiplicity is finite is used to build a characterization of finite-valued word transducers. The present work is thus a first step towards the characterization of finite-valued visibly pushdown transducers, which is a relevant issue as this model is incomparable with bottom-up tree transducers (see [8]).

The first problem we consider is the finiteness of the multiplicity of an automaton, *i.e.* does there exist $K \in \mathbb{N}$ such that the multiplicity is bounded by K. To solve this problem, we extend a characterization of finite state automata based on patterns to visibly pushdown automata. We also provide an algorithm to decide the presence of these patterns in polynomial time. The second class of problems asks whether the multiplicity of an automaton is bounded by K, where K is given. This problem can be considered under the hypothesis that K is part of the input, or is fixed. We show that the problem is EXPTIME-complete in the first case, and can be solved in polynomial time in the second one. Finally, we make a comparison of our results with existing results for the equivalent model of tree automata with weights on the semiring $(\mathbb{N}, +, \cdot)$. As this equivalence is effective, we discuss the consequences of our results in this context.

Definitions are given in Section 2. Comparisons with existing results for tree automata with multiplicities are drawn in Section 3. In Section 4, we give the characterization of $\mathbb{N}$-VPA with infinite multiplicity based on original patterns and the decision procedure associated. We study K-boundedness problems in Section 5, and conclude with an application of our results to tree automata in Section 6. Details of proofs and definitions about tree automata are omitted, and can be found in the appendix.

2 Definitions

2.1 Preliminaries

All over this paper, Σ denotes a finite alphabet partitioned into three disjoint sets Σ_c, Σ_r and Σ_ι, denoting respectively the *call*, *return* and *internal* alphabets. We denote by Σ^* the set of (finite) words over Σ and by ϵ the empty word. The length of a word u is denoted by $|u|$. The set of *well-nested* words Σ^*_{wn} is the smallest subset of Σ^* such that $\Sigma^*_\iota \subseteq \Sigma^*_{\mathsf{wn}}$ and for all $c \in \Sigma_c$, all $r \in \Sigma_r$, all $u, v \in \Sigma^*_{\mathsf{wn}}$, $cur \in \Sigma^*_{\mathsf{wn}}$ and $uv \in \Sigma^*_{\mathsf{wn}}$.

Let $u = \alpha_0\alpha_1 \cdots \alpha_{k-1} \in \Sigma^*$ be a word with $\alpha_i \in \Sigma$, for $0 \leq i \leq k-1$. Let $0 \leq i \leq j \leq |u|$, then $u_{i,j}$ denotes the word $\alpha_i \cdots \alpha_{j-1}$ if $i < j$, and the empty word if $i = j$. A position $i < |u|$ is a *pending call* if $\alpha_i \in \Sigma_c$ and for all $i < j \leq |u|$, $u_{i,j} \notin \Sigma^*_{\mathsf{wn}}$. The *height* of u at position i, denoted by $h_u(i)$, is the number of pending calls of $u_{0,i}$, i.e. $h_u(i) = |\{j \mid 0 \leq j < i \text{ and } \alpha_j \text{ is a pending call of } u_{0,i}\}|$. The *height* of u is the maximal height of all the positions of u: $h_u = \max_{0 \leq i \leq |u|} h_u(i)$. For instance, $h_{(crcrcc)} = h_{(ccrcrr)} = 2$.

2.2 Visibly Pushdown Automata with Multiplicities

Visibly pushdown automata [1] are a restriction of pushdown automata in which the stack behavior is imposed by the input word. On a call symbol, the VPA pushes a symbol onto the stack, on a return symbol, it must pop the top symbol of the stack and on an internal symbol, the stack remains unchanged.

Definition 1 (Visibly pushdown automata [1]). *A* visibly pushdown automaton *(*VPA*) over Σ is a tuple $A = (Q, \Gamma, \delta, Q_{in}, Q_f)$ where Q is a finite set of states, $Q_{in} \subseteq Q$ is the set of initial states, $Q_f \subseteq Q$ is the set of final states, Γ is a finite stack alphabet, $\delta = \delta_c \uplus \delta_r \uplus \delta_\iota$ is the set of transitions, with $\delta_c \subseteq Q \times \Sigma_c \times \Gamma \times Q$, $\delta_r \subseteq Q \times \Sigma_r \times \Gamma \times Q$, and $\delta_\iota \subseteq Q \times \Sigma_\iota \times Q$.*

Configuration - Run - Degree of ambiguity. A *configuration* of a VPA is a pair $(q, \sigma) \in Q \times \Gamma^*$ (where Γ^* denotes the set of finite words over Γ). We denote by $\perp$ the empty word on Γ. Initial (resp. final) configurations are configurations of the form $(q, \perp)$, with $q \in Q_{in}$ (resp. $q \in Q_f$).

A *run* of A on a sequence of transitions $\eta = \{t_i\}_{1 \leq i \leq k}$ from a configuration (q, σ) to a configuration (q', σ') over a word $u = \alpha_0 \dots \alpha_{k-1} \in \Sigma^*$ is a finite non-empty sequence $\rho = \{(q_i, \sigma_i)\}_{0 \leq i \leq k}$ such that $q_0 = q$, $\sigma_0 = \sigma$, $q_k = q'$, $\sigma_k = \sigma'$ and for each $1 \leq i \leq k$, $t_i = (q_{i-1}, \alpha_{i-1}, \gamma_i, q_i) \in \delta_c$ and $\sigma_i = \sigma_{i-1}\gamma_i$ or $t_i = (q_{i-1}, \alpha_i, \gamma_i, q_i) \in \delta_r$ and $\sigma_{i-1} = \sigma_i\gamma_i$, or $t_i = (q_{i-1}, \alpha_i, q_i) \in \delta_\iota$ and $\sigma_i = \sigma_{i-1}$. We say that the run is labeled by the word u and denote this run by $(q, \sigma) \xrightarrow{u} (q', \sigma')$. A run is *accepting* if it starts in an initial configuration and ends in a final configuration. The *degree of ambiguity* of A, denoted by $\mathsf{da}(A)$, is the maximal number of accepting runs for any possible input word.

Language. A word u is accepted by A if there exists an accepting run of A on u. The *language* of A, denoted by $\mathcal{L}(A)$, is the set of words accepted by A. Note that we require here to end up with an empty stack, this restriction implies that all accepted words are well-nested. Unlike [1], we do not consider returns on empty stack and unmatched calls. This assumption is done to avoid technical details but the general framework could be handled [1].

Trimmed. A configuration (q, σ) is *reachable* (resp. *co-reachable*) if there exists $u \in \Sigma^*$ and $q_0 \in Q_{in}$ (resp. $q_f \in Q_f$) such that $(q_0, \perp) \xrightarrow{u} (q, \sigma)$ (resp. such that $(q, \sigma) \xrightarrow{u} (q_f, \perp)$). A VPA A is *trimmed* if every reachable configuration is co-reachable, every co-reachable configuration is reachable and if every state of A belongs to a reachable configuration. In [4], we present a procedure which allows to trim a VPA and which preserves the set of accepting runs. We also prove that this procedure can be applied to the model of $\mathbb{N}$-VPA (see below).

Path. A path over a word $u \in \Sigma^*$ is a sequence of transitions $\eta = \{t_i\}_{1 \leq i \leq k}$ such that there exists a run on η labeled by the word u. Note that there may be different

[1] More precisely, given a general VPA A, one can build a VPA A' according to Definition 1 such that accepting runs of A' are in bijection with those of A. This can be achieved by adding self-loops on initial states that allow to push a special symbol (for the returns on empty stack) and self-loops on final states that allow to pop any symbols.

runs on the same path, differing in their initial configurations. The empty path (on the empty word ϵ) is denoted by η_ϵ. A path is said to be accepting whenever there exists an accepting run over it. Let η be a path over a word $u \neq \epsilon$, then there exist states p and q such that any run over η goes from a configuration (p, σ) to a configuration (q, σ') for some $\sigma, \sigma' \in \Gamma^*$. We then say that η goes from p to q, and write $\eta : p \xrightarrow{u} q$.

Lemma 1.

a. Let $u_i \in \Sigma^* \setminus \{\epsilon\}$ and $\eta_i : p_i \xrightarrow{u_i} q_i$ a path over u_i for $i \in \{1, 2, 3\}$ such that $u_1u_3, u_2 \in \Sigma^*_{\mathsf{wn}}$, and $\eta_1\eta_2\eta_3$ is a path. Then:
   - *for all $\eta'_2 : p_2 \xrightarrow{u'_2} q_2$ such that $u'_2 \in \Sigma^*_{\mathsf{wn}} \setminus \{\epsilon\}$, $\eta_1\eta'_2\eta_3$ is a path,*
   - *if $p_1 = q_1$ and $p_3 = q_3$, then $\eta_1^2\eta_2\eta_3^2$ is a path.*

b. Assume A is trimmed. For any family $(\eta_i)_{i\in I}$ of paths going from p to q on some well-nested word $u \neq \epsilon$, there exist two paths η', η'' such that for any $i \in I$, $\eta'\eta_i\eta''$ is an accepting path.

We introduce the model of VPA with multiplicities in $\mathbb{N}$ ($\mathbb{N}$-VPA for short), where transitions are labeled by positive integers:

Definition 2 ($\mathbb{N}$-VPA). *An $\mathbb{N}$-VPA is a pair $T = (A, \lambda)$ composed of a VPA $A = (Q, \Gamma, \delta, Q_{in}, Q_f)$ and a labeling function $\lambda : \delta \to \mathbb{N}_{>0}$.*

The notions of configurations, runs and paths are lifted from VPA to $\mathbb{N}$-VPA. We define the language of an $\mathbb{N}$-VPA $T = (A, \lambda)$ as the language of A.

Multiplicity. For each transition $t \in \delta$, $\lambda(t)$ is called the *multiplicity* of t. Let $\eta = \{t_i\}_{1\leq i\leq k}$ be a path of A over the word u and let $m_i = \lambda(t_i)$ for $1 \leq i \leq k$. The multiplicity of η denoted by $\langle\eta\rangle$ is $\prod_{1\leq i\leq k} m_i$. Let a word $u \neq \epsilon$, we write $p \xrightarrow{u|m} q$ when there exists a path over u from p to q with multiplicity m. The multiplicity of the empty path η_ϵ is 1.

We define the *multiplicity* of a run ρ, denoted by $\langle\rho\rangle$, as the one of its underlying path η. Let $u \in \mathcal{L}(T)$ be a word. The *multiplicity* of u, denoted by $\langle u\rangle$ is the sum of the multiplicities of the accepting runs for the word u. The *multiplicity* of an $\mathbb{N}$-VPA T, denoted by $\langle T\rangle$, is defined as $\langle T\rangle = \sup\{\langle u\rangle \mid u \in \mathcal{L}(T)\}$. Let $K \in \mathbb{N}$. We say that T is bounded by K if $\langle T\rangle \leq K$. We say that T is *finite* if we have $\langle T\rangle < +\infty$, and *infinite* otherwise. Note that the degree of ambiguity of a VPA is equal to the multiplicity of the corresponding $\mathbb{N}$-VPA where all the multiplicities of transitions are set to 1.

3 Relating Tree Automata and VPA

There is a strong relationship between words written over a partitioned alphabet and (un)ranked trees. This relationship extends to recognizers with VPA on one side and tree automata on the other side. A polynomial time construction from VPA to tree automata is presented in [1]. This latter construction preserves the language but not the computations; however, the construction can be slightly modified to guarantee the isomorphism of accepting computations [3]. Conversely, it is easy to encode ranked trees as well-nested visible words, and to build from a tree automaton a VPA accepting the encodings and preserving the accepting computations as well.

Note that preserving (accepting) computations implies that the degree of ambiguity of the encoded VPA and of the target tree automaton are the same.

Hence, one may now wonder whether this relationship extends to models with weights and what are the results known for weighted tree automata that carry over $\mathbb{N}$-VPA: this question is crucial as in one direction, it may be the case that problems we want to address could be solved thanks to this relationship and on the other direction, new results for $\mathbb{N}$-VPA may carry over weighted tree automata almost for free. Weighted tree automata [9] over the semiring $(\mathbb{N}, +, .)$ allow to encode $\mathbb{N}$-VPA: the weight of a node in a run is the product of the weight of its children multiplied by the one associated with the transition rule applied at this node, the weight of a tree being the sum of the weights of its accepting runs. Thanks to one-to-one isomorphism between the transitions of the $\mathbb{N}$-VPA and the ones of the tree automaton recognizing stack trees, weights are preserved by this translation. Conversely, when a (ranked) tree automaton is translated into a VPA, a transition rule for some symbol a of the tree automaton is encoded as two rules in the VPA (one for a call symbol $\langle a$, one for a return symbol $a\rangle$), the weight of the rule in the tree automaton being associated with one of the twos, the other one having multiplicity 1 (see Appendix C).

Let us briefly recap some known results for tree automata with weights/costs. In [14], (ranked) tree automata with polynomial costs are considered over several semirings. The main ingredient of these automata is that a polynomial over a semiring is attached to transitions : computing the cost of a node amounts to apply the polynomial with each variable x_i instanciated with the cost of the ith child. However, the result of the computation is the set of costs computed for each accepted run (no combination is made with the accepting computations over the same input tree). Finiteness and K-boundedness problems whose decidability issues are addressed relate to the finiteness and to the K-boundedness of this set of costs (shown to be in PTIME for many semirings and in particular, $(\mathbb{N}, +, .)$) and is thus different from the problems we consider here. These results are extended in [2] by considering more general semirings but without addressing complexity issues.

As already mentioned, the degree of ambiguity and the multiplicity of automata are related. In particular, finiteness or K-boundedness problems of the degree of ambiguity of tree automata provide lower bounds for the corresponding problems for $\mathbb{N}$-VPA.

However, the algorithms for finiteness of the degree of ambiguity [12] (deciding $\mathsf{DA} = \mathsf{da}(A) < +\infty$) in PTIME and of the cost of some tree automaton with costs [14] (deciding $\mathsf{MM} = \sup\{\langle\rho\rangle \mid \rho \text{ an accepting computation}\} < +\infty$) in PTIME can be combined to get a PTIME algorithm for finiteness of weighted tree automata, thanks to the following statement : $\max(\mathsf{DA}, \mathsf{MM}) \leq \langle A\rangle \leq \mathsf{DA} * \mathsf{MM}$. Thanks to the PTIME encoding of $\mathbb{N}$-VPA into weighted tree automata preserving the degree of ambiguity and the multiplicities of encoded computations, we obtain a PTIME algorithm for finiteness of $\mathbb{N}$-VPA. However, our approach provides a direct method based on VPA and a rather intuitive algorithm compared to [12,14]. Moreover, we will see in Section 6 that conversely, our approach leads to a new vision and a new and rather simple PTIME algorithm for finiteness of weighted tree automata over $(\mathbb{N}, +, .)$.

[14] also relates the degree of ambiguity and costs provided the use of multi dimensional cost automata. We believe that this may be extended to the computation of

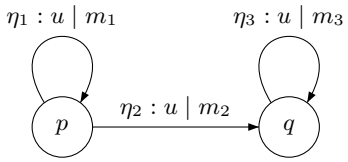
multiplicities. As pointed out in [14], this would yield an exponential time-complexity method to test K-boundedness, the algorithm being exponential in the dimension which is in this case the number of states of the tree automaton (we will show that this problem with the binary encoding of K being part of the input, for VPA and for tree automata is EXPTIME-hard). However, we will present a much simpler approach based on [5] to tackle this problem.

4 Characterization and decision of infinite $\mathbb{N}$-VPA

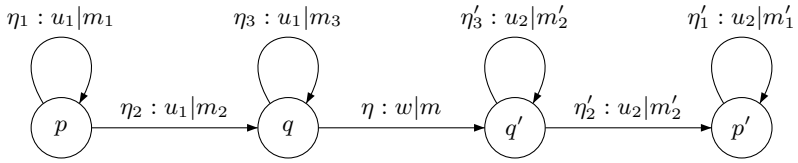
In this section, we give a characterization on $\mathbb{N}$-VPA ensuring their infiniteness by means of patterns. Then, based on this characterization, we devise a PTIME algorithm to solve the finiteness problem. All over this section, we assume a trimmed $\mathbb{N}$-VPA $T = (A, \lambda)$, with $A = (Q, \Gamma, \delta, Q_{in}, Q_f)$. We denote by n the cardinality of Q, and by L the value $\max\{\lambda(t) \mid t \in \delta\}$.

4.1 Characterization

We introduce the criteria depicted on Figures 1(a) and 1(b) which characterize infinite $\mathbb{N}$-VPA. Pattern of Figure 1(a) coincides with patterns for finite-state automata with multiplicities (see [15,7]). Pattern of Figure 1(b) is specific to the model of VPA. Intuitively, the loop over a well-nested word is splitted into two loops on words u_1 and u_2, such that the concatenation u_1u_2 is a well-nested word but u_1 is not well-nested. We say that T contains a pattern whenever there exist words in Σ^*, states of T and paths in T that fulfill all the conditions of the pattern. For instance, if we consider the pattern **(S1)**, we should find a word $u \in \Sigma^*_{\mathsf{wn}}$, two states $p, q \in Q$ (which may be equal), three paths $\eta_1 : p \xrightarrow{u|m_1} p$, $\eta_2 : p \xrightarrow{u|m_2} q$, $\eta_3 : q \xrightarrow{u|m_3} q$ such that $\eta_1\eta_2\eta_3$ is a path, and $m_1 > 1$ or $\eta_1 \neq \eta_2$. In these patterns, all words except w are necessarily non-empty. Note that these patterns also yield a characterization of infinite ambiguity by removing the disjunctions on multiplicities (conditions $m > 1$).



(a) **(S1)** Well-nested case: $u \in \Sigma^*_{\mathsf{wn}}$. $\eta_1\eta_2\eta_3$ is a path and $m_1 > 1$ or $\eta_1 \neq \eta_2$.



(b) **(S2)** Matched loops case: $w \in \Sigma^*_{\mathsf{wn}}$, $u_1u_2 \in \Sigma^*_{\mathsf{wn}}$, and $u_1 \notin \Sigma^*_{\mathsf{wn}}$. $\eta_1\eta_2\eta_3\eta\eta'_3\eta'_2\eta'_1$ is a path, and either ($m_1 > 1$ or $m'_1 > 1$), or ($\eta_1 \neq \eta_2$ or $\eta'_1 \neq \eta'_2$).

Fig. 1. Patterns characterizing infinite multiplicity.

We will show in this section that these criteria characterize infinite $\mathbb{N}$-VPA:

Theorem 1. *Let T be an $\mathbb{N}$-VPA. T is infinite if and only if T complies with one of the criteria **(S1)** and **(S2)**.*

To prove this result, we first show that if we have one of the criteria then the multiplicity is infinite. In a second part we show that if the multiplicity is infinite, then the $\mathbb{N}$-VPA complies with one of the criteria.

Lemma 2. *Let T be an $\mathbb{N}$-VPA. If T complies with **(S1)** or **(S2)**, then T is infinite.*

Proof (Sketch). We sketch the proof for criterion **(S1)**, the case of **(S2)** being similar. Let $u \in \Sigma^*_{\mathsf{wn}}$ and η_1, η_2, η_3 be paths selected according to pattern **(S1)**. We first suppose that condition $m_1 > 1$ holds. As η_1 is a path going from p to p and $u \in \Sigma^*_{\mathsf{wn}}$, η_1^2 is also a path from p to p. By applying iteratively Lemma 1.a, we can consider path η_1^i whose multiplicity $\langle \eta_1^i \rangle = m_1^i$ grows to infinity when i tends to $+\infty$. As T is trimmed, by Lemma 1.b, this gives accepting paths with multiplicity growing to infinity. Consider now the case where the condition $\eta_1 \neq \eta_2$ holds. Let $k \in \mathbb{N}_{>0}$, and $i, j \in \mathbb{N}$ such that $i + j = k - 1$. As $\eta_1\eta_2\eta_3$ is a path and $u \in \Sigma^*_{\mathsf{wn}}$, by Lemma 1.a, the path $\eta_1^i\eta_2\eta_3^j$ is a path over the word u^k. Moreover, as $\eta_1 \neq \eta_2$, all these paths are different when i, j range over the set of integers such that $i + j = k - 1$. When k tends to infinity, we obtain an arbitrarily large number of paths over u^k going from p to q. As T is trimmed, by Lemma 1.b, this gives arbitrarily many accepting paths over a same word. □

The proof of the converse (an infinite multiplicity implies the presence of one of the criteria) relies on the two technical Lemmas 4 and 5 which we present intuitively. To state these lemmas, we define the constant $N = (n^2L)^2|\Gamma|$ and the function $\psi : \mathbb{N} \to \mathbb{N}$ as $\psi(z) = n(Nz)^{2^n}$. Pattern **(S1)** allows to increase the multiplicity along a well-nested word. Lemma 4 states that if T does not comply with **(S1)**, then a well-nested word u whose multiplicity is greater than $\psi(l)$ has a well-nested subword v whose multiplicity is greater than l, and such that $h_u > h_v$. Then, Lemma 5 applies iteratively Lemma 4 to prove that a word with large multiplicity has a large height, and hence allows to find pattern **(S2)**, using a vertical pumping.

Let $u \in \Sigma^*_{\mathsf{wn}}$ and $k = |u|$. Given two positions i, j such that $0 \leq i \leq j \leq k$ and $u_{i,j} \in \Sigma^*_{\mathsf{wn}}$, we define a matrix, denoted by $\mathsf{induced}^u_{i,j}$, representing intuitively how the multiplicities of runs are modified by the subword $u_{i,j}$. Formally, $\mathsf{induced}^u_{i,j}$ is an element of $\mathbb{N}^{Q\times Q}$, and for $p, q \in Q$, we let $\mathsf{induced}^u_{i,j}(p, q)$ be the sum of the multiplicities of the paths $\eta : p \xrightarrow{u_{i,j}} q$ of T for which there exist η_1 path on $u_{0,i}$, η_2 path on $u_{j,k}$ such that $\eta_1\eta\eta_2$ is an accepting path on u.

Finally, we also define $s^u_{i,j} \in \mathbb{N}$ as $s^u_{i,j} = \sum_{p,q\in Q} \mathsf{induced}^u_{i,j}(p, q)$. We have:

Lemma 3. *Let $u \in \Sigma^*_{\mathsf{wn}}$ and three positions i, j, k such that $0 \leq i \leq j \leq k \leq |u|$ and $u_{i,j}, u_{j,k}, u_{i,k} \in \Sigma^*_{\mathsf{wn}}$. Then we have $\mathsf{induced}^u_{i,k} = \mathsf{induced}^u_{i,j} \times \mathsf{induced}^u_{j,k}$, $s^u_{i,j} \leq s^u_{i,k}$, and $s^u_{i,k} \leq s^u_{i,j}.s^u_{j,k}$.*

Lemma 4. *We suppose that T is infinite but T does not comply with **(S1)**. Let $u \in \mathcal{L}(T)$, $l \in \mathbb{N}_{>0}$ and x, y be two positions such that $0 \leq x \leq y \leq |u|$, $u_{x,y} \in \Sigma^*_{\mathsf{wn}}$ and $s^u_{x,y} \geq \psi(l)$. Then there exist two positions $x < x' \leq y' < y$ such that $u_{x',y'} \in \Sigma^*_{\mathsf{wn}}$, $h_u(x') = h_u(x) + 1$ and $s^u_{x',y'} \geq l$.*

Proof (Sketch). The proof is based on a pumping on positions in the set $P = \{i \in \mathbb{N}_{>0} \mid x \leq i \leq y \wedge u_{x,i} \in \Sigma^*_{\mathsf{wn}}\}$. This approach is similar to that used in [7] for automata on words. For each $i \in P$, we define $r_i = s^u_{x,i}$ and $X_i = \{q \in Q \mid \mathsf{induced}^u_{x,i}(p,q) > 0 \text{ for some } p\}$. Intuitively, r_i corresponds to the multiplicity associated with the well-nested subword $u_{x,i}$ and X_i is the set of states that can be reached after this subword (along an accepting path over u). For any $i < j$, $i, j \in P$, we have thanks to Lemma 3, $r_i \leq r_j$ and $r_j \leq r_i \times s^u_{i,j}$.

Suppose, for the sake of contradiction, that for any two *consecutive* indices $i < j$ in P, we have $s^u_{i,j} < Nl$. Using the hypothesis $r_y = s^u_{x,y} \geq \psi(l)$ and the definition of ψ, we can prove that this entails that there exist two positions $i < j$ in the set P such that $r_i < r_j$ and $X_i = X_j$. Let $v = u_{i,j}$ and $X = X_i$. We define a multigraph $\mathcal{X} = (X, E)$ where $E \subseteq X \times X \times \mathbb{N}$ is defined as follows: $\forall p, q \in X, m \in \mathbb{N}$, for each path $\eta : p \xrightarrow{v|m} q$ such that $\eta'\eta\eta''$ is an accepting path on u for some paths η' on word $u_{0,i}$ and η'' on word $u_{j,|u|}$, we construct an edge $p \xrightarrow{m} q \in E$. Thanks to property $r_i < r_j$, we show that either there is a vertex with two outgoing edges, or $\mathcal{X}$ is composed of disjoint loops and contains an edge with label $m > 1$. In the former case, we prove that T contains pattern **(S1)** with property $\eta_1 \neq \eta_2$ while in the latter case, we prove it contains **(S1)** with property $m_1 > 1$. This contradicts our hypothesis on T.

Hence, we have proven that there exist two indices $i < j$ in P such that $s^u_{i,j} \geq Nl$. Then, we can extract from i and j the two expected indices x' and y'. □

Lemma 5. *Let T be an $\mathbb{N}$-VPA. If T is infinite, then T complies with one of the two criteria **(S1)** and **(S2)**.*

Proof (Sketch). Suppose that T is infinite but does not comply with **(S1)**, and prove it complies with **(S2)**. Let $u \in \mathcal{L}(T)$ be a word such that $\langle u \rangle \geq \psi^H(1)$ where $H = 2^{n^2}$ and $\psi^{h+1} = \psi \circ \psi^h$, for $h \in \mathbb{N}$. By applying Lemma 4 iteratively, we define a sequence of length greater than H of couples of positions $\chi_i = (x_i, y_i)$ of u. These couples represent well-nested subwords of u, which are recursively embedded. In addition, their multiplicities $s^u_{\chi_i}$ are strictly decreasing. We then proceed with a pumping argument similar to the one done in the proof of Lemma 4, and exhibit the pattern **(S2)**. □

4.2 Decidability of finiteness

We show in this part how to decide in PTIME the presence of one of the patterns.

The algorithm (Figure 2) uses four bunches of inference rules applied as a saturation procedure: the first bunch builds a set $\mathcal{S}_0$ of pairs (p, q) such that there exists a path over a well-nested word from p to q. The second bunch builds a set $\mathcal{S}_1$ of tuples composed of 6 states and a Boolean, which allows to decide the presence of the pattern **(S1)**. The 6 states represent the source and the target of 3 paths over the same well-nested word and the Boolean retains an information about a multiplicity greater than 1 or the fact that different paths are considered. The third bunch builds a set $\mathcal{S}_2$ of tuples composed of 12 states and a Boolean which allows to decide the presence of pattern **(S2)**. This construction is based on $\mathcal{S}_1$: the states aim to identify two sets of 3 paths over two words u_1 and u_2, such that the second set pops the stack pushed by the first set, ensuring that $u_1u_2 \in \Sigma^*_{\mathsf{wn}}$. The information stored in the Boolean depends on one of the sets. Finally,

the last bunch builds a set $\mathcal{S}_3$ which ensures that some tuple built in $\mathcal{S}_1$ forms the pattern **(S1)** in the rule 4.1, or that some tuple built in $\mathcal{S}_2$ represents the pattern **(S2)**, which in addition are connected through a well-nested word (condition over $\mathcal{S}_0$) in the rule 4.2.

Proposition 1. *For any* $\mathbb{N}$-VPA T, $(\top) \in \mathcal{S}_3$ *iff* T *is infinite.*

Theorem 2. *Finiteness for* $\mathbb{N}$-VPA *is in* PTIME.

$$\frac{p \in Q}{(p,p) \in \mathcal{S}_0}\ (1.1) \qquad \frac{(p,a,q) \in \delta_\iota}{(p,q) \in \mathcal{S}_0}\ (1.2) \qquad \frac{(p,q) \in \mathcal{S}_0, (q,q') \in \mathcal{S}_0}{(p,q') \in \mathcal{S}_0}\ (1.3)$$

$$\frac{(p,q) \in \mathcal{S}_0, (p',c,\gamma,p) \in \delta_c, (q,r,\gamma,q') \in \delta_r}{(p',q') \in \mathcal{S}_0}\ (1.4)$$

$$\frac{p_i \in Q \ \text{ for all } i \in \{1,2,3\}}{(p_1,p_1,p_2,p_2,p_3,p_3,\bot) \in \mathcal{S}_1}\ (2.1) \qquad \frac{t_i = (p_i,a,p_i') \in \delta_\iota \ \text{ for all } i \in \{1,2,3\}}{(p_1,p_1',p_2,p_2',p_3,p_3',(t_1 \neq t_2 \vee \varphi_1)) \in \mathcal{S}_1}\ (2.2)$$

$$\frac{(p_1,q_1,p_2,q_2,p_3,q_3,B) \in \mathcal{S}_1, (q_1,q_1',q_2,q_2',q_3,q_3',B') \in \mathcal{S}_1}{(p_1,q_1',p_2,q_2',p_3,q_3',B \vee B') \in \mathcal{S}_1}\ (2.3)$$

$$\frac{\begin{array}{c}(p_1,q_1,p_2,q_2,p_3,q_3,B) \in \mathcal{S}_1,\\ t_i = (p_i',c,\gamma_i,p_i) \in \delta_c, t_i' = (q_i,r,\gamma_i,q_i') \in \delta_r \ \text{ for all } i \in \{1,2,3\}\end{array}}{(p_1',q_1',p_2',q_2',p_3',q_3',B \vee (t_1 \neq t_2 \vee t_1' \neq t_2' \vee \varphi_1' \vee \varphi_1)) \in \mathcal{S}_1}\ (2.4)$$

$$\frac{(p_1,q_1,p_2,q_2,p_3,q_3,B) \in \mathcal{S}_1, (q_3',p_3',q_2',p_2',q_1',p_1',B') \in \mathcal{S}_1}{(p_1,q_1,p_2,q_2,p_3,q_3,q_3',p_3',q_2',p_2',q_1',p_1',B \vee B') \in \mathcal{S}_2}\ (3.1)$$

$$\frac{t_i = (p_i',c,\gamma_i,p_i) \in \delta_c, \ \ t_i' = (q_i,r,\gamma_i,q_i') \in \delta_r, \ \ \text{for all } i \in \{1,2,3\}}{(p_1',p_1,p_2',p_2,p_3',p_3,q_3,q_3',q_2,q_2',q_1,q_1',t_1 \neq t_2 \vee t_1' \neq t_2' \vee \varphi_1 \vee \varphi_1') \in \mathcal{S}_2}\ (3.2)$$

$$\frac{\begin{array}{c}(p_1,p_1',p_2,p_2',p_3,p_3',q_3',q_3,q_2',q_2,q_1',q_1,B) \in \mathcal{S}_2,\\ (p_1'',p_1,p_2'',p_2,p_3'',p_3,q_3,q_3'',q_2,q_2'',q_1,q_1'',B') \in \mathcal{S}_2\end{array}}{(p_1'',p_1',p_2'',p_2',p_3'',p_3',q_3',q_3'',q_2',q_2'',q_1',q_1'',B \vee B') \in \mathcal{S}_2}\ (3.3)$$

$$\frac{(p,p,p,q,q,q,\top) \in \mathcal{S}_1}{(\top) \in \mathcal{S}_3}\ (4.1) \qquad \frac{\begin{array}{c}(q,q') \in \mathcal{S}_0,\\ (p,p,p,q,q,q,q',q',q',p',p',p',\top) \in \mathcal{S}_2\end{array}}{(\top) \in \mathcal{S}_3}\ (4.2)$$

with $\varphi_1 = \lambda(t_1) > 1$ and $\varphi_1' = \lambda(t_1') > 1$

Fig. 2. Inference rules for deciding finiteness

5 Finite bounds for $\mathbb{N}$-VPA

5.1 Deciding K-bounded multiplicity

We consider a trimmed $\mathbb{N}$-VPA $T = (A, \lambda)$, with $A = (Q, \Gamma, \delta, Q_{in}, Q_f)$ and an integer $K \in \mathbb{N}_{>0}$ represented in binary and describe an algorithm to decide whether $\langle T \rangle < K$.

The procedure we describe builds a set $\mathcal{M}$ of $n \times n$ integer matrices by saturation, where rows and columns of matrices are indexed by states of A. The semantics of a

matrix $M \in \mathcal{M}$ can be understood as follows: there exists a word $u \in \Sigma^*_{\mathsf{wn}}$ such that, for any $p, q \in Q$, the entry $M(p,q)$ is equal to the sum of the multiplicities of paths $p \xrightarrow{u} q$. We have then $\langle u \rangle = \sum_{q_i \in Q_{in}, q_f \in Q_f} M(q_i, q_f)$. For an $n \times n$ integer matrix M, we denote by $\langle M \rangle$ the value $\sum_{q_i \in Q_{in}, q_f \in Q_f} M(q_i, q_f)$.

The algorithm proceeds by building such matrices for well-nested words of increasing lengths. It starts with internal words of length 1, and then extends words either by concatenation, or by adding a matching pair of call/return symbols.

We introduce the following notations: let M_ϵ be the identity matrix. Let $a \in \Sigma_\iota$, then M_a is the matrix defined by $M_a(p,q) = \lambda(t)$ if there exists $t = (p, a, q) \in \delta_\iota$, and $M_a(p,q) = 0$ otherwise. Let $\gamma \in \Gamma$ and let $c \in \Sigma_c$ (resp. $r \in \Sigma_r$), then $M_{c,\gamma}$ (resp. $M_{r,\gamma}$) is the matrix defined by $M_{c,\gamma}(p,q) = \lambda(t)$ if there exists $t = (p, c, \gamma, q) \in \delta_c$, and $M_{c,\gamma}(p,q) = 0$ otherwise (and similarly for matrix $M_{r,\gamma}$).

Finally, we introduce the operator $\mathsf{Extra}_K : \mathbb{N} \to \mathbb{N}$ defined by $\mathsf{Extra}_K(z) = z$ if $z \leq K$, and $\mathsf{Extra}_K(z) = K$ otherwise. This operator is naturally extended to integer matrices. Our algorithm is presented as Algorithm 1.

Algorithm 1 Decision of the K-boundedness of an $\mathbb{N}$-VPA

Require: An $\mathbb{N}$-VPA T and $K \in \mathbb{N}_{>0}$
1: $\mathcal{M} \leftarrow \{\mathsf{Extra}_K(M_a) \mid a \in \Sigma_\iota\} \cup \{M_\epsilon\}$
2: $\mathcal{M}' \leftarrow \emptyset$
3: **repeat**
4: $\quad \mathcal{M} \leftarrow \mathcal{M} \cup \mathcal{M}'$
5: $\quad$ **if** $\exists M \in \mathcal{M}$ such that $\langle M \rangle \geq K$ **then**
6: $\quad\quad$ **return** `false`
7: $\quad$ **end if**
8: $\quad \mathcal{M}' \leftarrow \{\mathsf{Extra}_K(M_1.M_2) \mid M_1, M_2 \in \mathcal{M}\} \cup$
$\quad\quad\quad \{\mathsf{Extra}_K(\sum_{\gamma \in \Gamma} M_{c,\gamma}.M.M_{r,\gamma}) \mid M \in \mathcal{M}, c \in \Sigma_c, r \in \Sigma_r\}$
9: **until** $\mathcal{M}' \cup \mathcal{M} = \mathcal{M}$
10: **return** `true`

Theorem 3. *Given an* $\mathbb{N}$-VPA T *and* $K \in \mathbb{N}_{>0}$*, the problem of determining whether* $\langle T \rangle < K$ *is* EXPTIME*-complete.*

This complexity should be compared with that of determining whether the ambiguity of a finite state automaton is less than K which is known to be PSPACE-complete [5].

Proof (Sketch). The EXPTIME membership follows from the fact that all the matrices built are $n \times n$ matrices whose entries are bounded by K. For the hardness, we can proceed to a reduction from the emptiness of the intersection of K deterministic bottom-up tree automata [6]. One can first consider the tree automaton obtained as the disjoint union of these K automata. Then one can turn this tree automaton into a VPA accepting the encodings of the trees as well-nested words, and with an isomorphic set of accepting runs. Considering this VPA as an $\mathbb{N}$-VPA T (each multiplicity is set to 1), one can show that the intersection of the K deterministic tree automata is empty iff $\langle T \rangle < K$. □

Computing the multiplicity of a finite $\mathbb{N}$-VPA Consider now, given a finite $\mathbb{N}$-VPA, the problem consisting in computing its multiplicity. We derive from the previous algorithm a procedure solving this problem. The procedure simply explores as before the set of

matrices, without using the operator Extra_K, until saturation of the set of matrices. The termination of the algorithm relies on the fact that T is finite and trimmed. Indeed, this entails that coefficients computed are all bounded by $\langle T \rangle$. In particular, this proves that the number of matrices built is bounded by $\langle T \rangle^{n^2}$, and:

Theorem 4. *For all finite* $\mathbb{N}$-VPA T*, the value of* $\langle T \rangle$ *can be computed in time* $\langle T \rangle^{O(n^2)}$.

5.2 Deciding K-bounded multiplicity (for a fixed K)

As a final result in this section, we investigate the K-bounded multiplicity problem for which the input is only an $\mathbb{N}$-VPA T and we ask whether $\langle T \rangle < K$. The algorithm from Section 5.1 shows that this problem for a fixed K can be solved in exponential time; however, by adapting the approach used in [13] for ambiguity of tree automata,

Theorem 5. *Fix* $K \in \mathbb{N}_{>0}$*. For an* $\mathbb{N}$-VPA T*, deciding whether* $\langle T \rangle < K$ *is in* PTIME.

Proof. We consider the family of VPA's $(A_i)_{1 \leq i \leq K}$ such that A_i accepts words from $\mathcal{L}(T)$ having a multiplicity greater than K. More precisely, A_i is a VPA that accepts words u such that there are i different accepting runs $\rho_1, \ldots, \rho_i$ of T over u verifying $\sum_{1 \leq j \leq i} \langle \rho_j \rangle \geq K$. Therefore, A_i simulates in parallel i runs of T over the same word, and for each of them keeps track of the current multiplicity in its states by computing up to K. More precisely, for $T = (A, \lambda)$, with $A = (Q, \Gamma, \delta, Q_{in}, Q_f)$, we define A_i as $(Q_i, \Gamma_i, \delta_i, Q^i_{in}, Q^i_f)$ where $Q_i = (Q \times [1..K])^i \times \mathbb{B}^{i \times i}$, $\Gamma_i = (\Gamma)^i$, $Q^i_{in} = (Q_{in} \times \{1\})^i \times \{\mathbf{0}_{\mathbb{B}^{i \times i}}\}$, $Q^i_f = \{((q_1, m_1), \ldots, (q_i, m_i)) \times \{\overline{\mathbf{Id}}_{\mathbb{B}^{i \times i}}\} \mid (q_1, \ldots, q_i) \in (Q_f)^i, (\sum_{1 \leq j \leq i} m_j) \geq K\}$. The element of $\mathbb{B}^{i \times i}$, which is the set of the $i \times i$ square matrices of Booleans, is used to store whether the runs are distinct. $\mathbf{0}_{\mathbb{B}^{i \times i}}$ (resp. $\overline{\mathbf{Id}}_{\mathbb{B}^{i \times i}}$) is the matrix containing only false values (resp. only true values except on the diagonal which is set to false), i.e. all runs are equal (resp. distinct). Let $\delta_i = \delta^c_i \uplus \delta^\iota_i \uplus \delta^\tau_i$ where

$$\delta^c_i = \left\{ \begin{array}{c} (((q_1, m_1), \ldots, (q_i, m_i), M), \\ c, (\gamma_1, \ldots, \gamma_i), \\ ((q'_1, m'_1), \ldots, (q'_i, m'_i), M')) \end{array} \middle| \begin{array}{c} c \in \Sigma_c, \text{for all } 1 \leq j, l \leq i, \\ t_j = (q_j, c, \gamma_j, q'_j), t_l = (q_l, c, \gamma_l, q'_l) \in \delta_c \\ m'_j = \mathsf{Extra}_K(\lambda((q_j, c, \gamma_j, q'_j)) * m_j) \text{ and} \\ M'(j,l) = M(j,l) \vee (t_j \neq t_l) \end{array} \right\}$$

δ^ι_i and δ^τ_i are defined similarly. It is obvious that each A_i can be built in polynomial time in $|T|$. Finally, we test in polynomial time for emptiness each of the K VPA A_i. □

6 Back to Trees

Considering the polynomial encoding of (weighted) tree automata into VPA (with multiplicities), we can deduce the two following results:

1. Determining whether the ambiguity of a tree automaton A is less than K, when A and the binary encoding of K are part of the input, is EXPTIME-complete.
2. We exhibit a simple pattern characterizing infinite weighted tree automata over $\mathbb{N}$, which can be decided in PTIME. Moreover, it turns out to be the disjunction of a pattern for infinite ambiguity, and one for infinite cost (in the sense of [14]).

Point 1 should be compared with the PTIME complexity of this problem when K is fixed (see [13]). Regarding point 2, we claim (see Figure 3):

> a weighted tree automaton T over $\mathbb{N}$ is infinite iff there exists a one-hole context C and computations φ_i for $i \in \{1,2,3\}$ of T over C such that $\varphi_1 : p \xrightarrow{C} p$, $\varphi_2 : p \xrightarrow{C} q$, $\varphi_3 : q \xrightarrow{C} q$ for some $p, q \in Q$ verifying $\langle \varphi_1 \rangle > 1$ or $\varphi_1 \neq \varphi_2$.

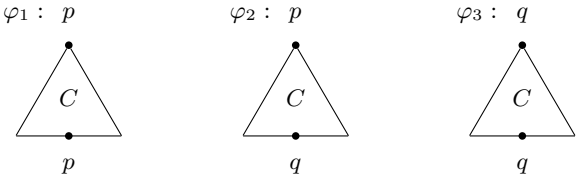


Fig. 3. Patterns for infinite weighted tree automata

We can then derive a PTIME algorithm for weighted tree automata rather similar to the one we proposed for $\mathbb{N}$-VPA (see Appendix C.3).

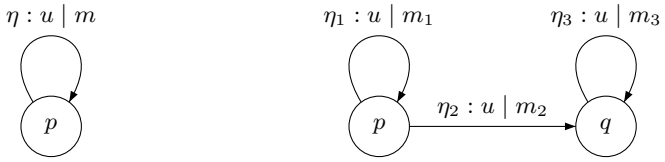
References

1. R. Alur and P. Madhusudan. Visibly pushdown languages. In *Proc. STOC '04*, pages 202–211, 2004.
2. B. Borchardt, Z. Fülöp, Z. Gazdag, and A. Maletti. Bounds for tree automata with polynomial costs. *Journal of Automata, Languages and Combinatorics*, 10(2/3):107–157, 2005.
3. M. Caralp. Automates à pile visible : ambiguité et valuation. Master's thesis, Aix-Marseille Université, 2011.
4. M. Caralp, P.-A. Reynier, and J.-M. Talbot. A polynomial procedure for trimming visibly pushdown automata. Technical Report hal-00606778, HAL, CNRS, France, 2011.
5. T.-H. Chan and O. H. Ibarra. On the finite-valuedness problem for sequential machines. *Theoretical Computer Science 23*, pages 95–101, 1983.
6. H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications, 2008.
7. R. De Souza. *Étude structurelle des transducteurs de norme bornée*. PhD thesis, ENST, France, 2008.
8. E. Filiot, J.-F. Raskin, P.-A. R. Reynier, F. Servais, and J.-M. Talbot. Properties of visibly pushdown transducers. In *Proc. MFCS'10*, volume 6281 of *LNCS*, pages 355–367. Springer, 2010.
9. Z. Fülöp and H. Vogler. *Handbook of Weighted Automata*, chapter Weighted Tree Automata and Tree Transducers. Springer, 2009.
10. J. Sakarovitch. *Elements of Automata Theory*. Cambridge University Press, 2009.
11. J. Sakarovitch and R. de Souza. On the decidability of bounded valuedness for transducers. In *Proc. MFCS'11*, volume 5162 of *LNCS*, pages 588–600. Springer, 2008.
12. H. Seidl. On the finite degree of ambiguity of finite tree automata. *Acta Inf.*, 26(6):527–542, 1989.
13. H. Seidl. Deciding equivalence of finite tree automata. *SIAM J. Comput.*, 19(3):424–437, 1990.
14. H. Seidl. Finite tree automata with cost functions. *Theor. Comput. Sci.*, 126(1):113–142, 1994.
15. A. Weber and H. Seidl. On the degree of ambiguity of finite automata. *Theor. Comput. Sci.*, 88(2):325–349, 1991.

A Proofs of Section 4

A.1 Equivalent patterns

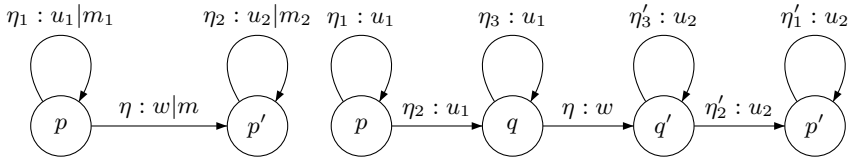
We present in an equivalent way the two patterns **(S1)** and **(S2)**. It is not difficult to verify that **(S1)** (resp. **(S2)**) is equivalent to **(S1.a)** or **(S1.b)** (resp. **(S2.a)** or **(S2.b)**).



(a) Heavy cycle: $m > 1$.

(b) Dumbbell: $\eta_1\eta_2\eta_3$ is a path and $\eta_1 \neq \eta_2$.

Fig. 4. (S1) Well-nested case: $u \in \Sigma^*_{\mathsf{wn}}$.



(a) Heavy cycle: $\eta_1\eta\eta_2$ is a path and ($m_1 > 1$ or $m_2 > 1$).

(b) Dumbbell: $\eta_1\eta_2\eta_3\eta\eta'_3\eta'_2\eta'_1$ is a path and ($\eta_1 \neq \eta_2$ or $\eta'_1 \neq \eta'_2$). Multiplicities have been omitted for readability.

Fig. 5. (S2) Matched loops case: $w \in \Sigma^*_{\mathsf{wn}}$, $u_1u_2 \in \Sigma^*_{\mathsf{wn}}$, and $u_1 \notin \Sigma^*_{\mathsf{wn}}$.

A.2 Proof of Lemma 1

Proof. a. Let $u_i \in \Sigma^* \setminus \{\epsilon\}$ and $\eta_i = p_i \xrightarrow{u_i} q_i$ be a path of T for $i \in \{1, 2, 3\}$ such that $u_1u_3, u_2 \in \Sigma^*_{\mathsf{wn}}$ and $\eta_1\eta_2\eta_3$ is a path of T. By definition of paths and since $u_1u_3, u_2 \in \Sigma^*_{\mathsf{wn}}$, there exist three runs $\rho_1 = (p_1, \bot) \xrightarrow{u_1} (q_1, \sigma)$, $\rho_2 = (p_2, \bot) \xrightarrow{u_2} (q_2, \bot)$ and $\rho_3 = (p_3, \sigma') \xrightarrow{u_3} (q_3, \bot)$ in T for $\sigma, \sigma' \in \Gamma^*$. Since $\eta_1\eta_2\eta_3$ is a path, there exists some $\sigma'' \in \Gamma^*$ such that $(p_1, \sigma'') \xrightarrow{u_1} (q_1, \sigma''\sigma) \xrightarrow{u_2} (q_2, \sigma''\sigma') \xrightarrow{u_3} (q_3, \sigma'')$ is a run of T. Observe that $u_2 \in \Sigma^*_{\mathsf{wn}}$, thus we have $\sigma''\sigma = \sigma''\sigma'$ and thus $\sigma = \sigma'$. It follows that for any path $\eta'_2 = p_2 \xrightarrow{u'_2} q_2$ with $u'_2 \in \Sigma^*_{\mathsf{wn}}$, there exists a run $\rho'_2 = (p_2, \bot) \xrightarrow{u'_2} (q_2, \bot)$ in T over η'_2. Thus, there is the run $(p_1, \sigma'') \xrightarrow{u_1} (q_1, \sigma''\sigma) \xrightarrow{u'_2} (q_2, \sigma''\sigma) \xrightarrow{u_3} (q_3, \sigma'')$ in T over $\eta_1\eta'_2\eta_3$ which is then a path of T.
We consider now that $p_1 = q_1$ and $p_3 = q_3$. Observe that since $\rho_1 = (p_1, \bot) \xrightarrow{u_1} (p_1, \sigma)$ and $\rho_3 = (p_3, \sigma) \xrightarrow{u_3} (p_3, \bot)$ are runs of T with $u_1u_3 \in \Sigma^*_{\mathsf{wn}}$, ρ^2_1 and ρ^2_3 are runs of T. Since $u_2 \in \Sigma^*_{\mathsf{wn}}$, there is a run $\rho_2 = (p_2, \bot) \xrightarrow{u_2} (q_2, \bot)$ in T, and thus $\rho'_2 = (p_2, \sigma\sigma) \xrightarrow{u_2} (q_2, \sigma\sigma)$ is also a run in T. Then $\eta^2_1\eta_2\eta^2_3$ is a path of T.

b. Let $(\eta_i)_{i\in I}$ be a family of paths going from p to q on some well-nested word $u \neq \epsilon$. Since T is trimmed, there exists $\sigma \in \Sigma^*$ such that (p, σ) is reachable. Let $i \in I$, and a run $(p, \perp) \xrightarrow{u} (q, \perp)$ over the well-nested word u. Then, configuration (q, σ) is reachable and thus, as T is trimmed, also co-reachable. Let ρ be a run which leads from a configuration $(q_{in}, \perp)$ (with $q_{in} \in Q_{in}$) to the configuration (p, σ) and ρ' a run which leads from the configuration (q, σ) to a configuration $(q_f, \perp)$ (with $q_f \in Q_f$). Let η (resp. η') be the path underlying run ρ (resp. ρ'). Then, for any $i \in I$, the concatenation $\eta\eta_i\eta'$ is an accepting path of T. □

A.3 Proof of Lemma 2

Proof. **Case 1** T complies with **(S1.a)**

Let $u \in \Sigma^*_{\mathsf{wn}}$ and η be a path selected according to pattern **(S1.a)**. As η is a path going from p to p and $u \in \Sigma^*_{\mathsf{wn}}$, η^2 is also a path from p to p. By applying iteratively Lemma 1.a, we can consider path η^i whose multiplicity $\langle \eta^i \rangle = m^i$ grows to infinity when i tends to $+\infty$. As T is trimmed, by Lemma 1.b, this gives accepting paths with multiplicity growing to infinity.

Case 2 T complies with **(S1.b)**

Let $u \in \Sigma^*_{\mathsf{wn}}$ and η_1, η_2, η_3 be paths selected according to pattern **(S1.b)**. Let $k \in \mathbb{N}_{>0}$, and $i, j \in \mathbb{N}$ such that $i + j = k - 1$. As $\eta_1\eta_2\eta_3$ is a path and $u \in \Sigma^*_{\mathsf{wn}}$, by Lemma 1.a, the path $\eta_1^i\eta_2\eta_3^j$ is a path over the word u^k. Moreover, as $\eta_1 \neq \eta_2$, all these paths are different when i, j range over the set of integers such that $i + j = k - 1$. When k tends to infinity, we obtain an arbitrarily large number of paths over u^k going from p to q. As T is trimmed, by Lemma 1.b, this gives arbitrarily many accepting paths over a same word.

Case 3 T complies with **(S2.a)**

Let $u_1u_2, w \in \Sigma^*_{\mathsf{wn}}$ and η_1, η, η_2 be paths selected according to the criterion **(S2.a)**. By applying iteratively Lemma 1.a, we can consider path $\eta_1^i\eta\eta_2^i$ whose multiplicity $m_1^i m m_2^i$ grows to infinity when i tends to $+\infty$. As T is trimmed, by Lemma 1.b, this gives accepting paths with multiplicity growing to infinity.

Case 4 T complies with **(S2.b)**

Let $u_1u_2, w \in \Sigma^*_{\mathsf{wn}}$ and $\eta_1, \eta_2, \eta_3, \eta, \eta'_3, \eta'_2, \eta'_1$ be paths selected according to the criterion **(S2.b)**. By applying iteratively Lemma 1.a, we can consider path $\eta_3^j\eta\eta_3'^j$ for any $j \in \mathbb{N}_{>0}$. As path $\eta_3^j\eta\eta_3'^j$ goes from q to q', Lemma 1.a entails that $\eta_2\eta_3^j\eta\eta_3'^j\eta'_2$ is a path. Then, the second point of Lemma 1.a entails that $\eta_1^i\eta_2\eta_3^j\eta\eta_3'^j$ $\eta'_2\eta_1'^i$ is a path for any $i \in \mathbb{N}_{>0}$. Let $k \in \mathbb{N}_{>0}$, one can observe that for any i, j such that $i + j = k - 1$, the path $\eta_1^i\eta_2\eta_3^j\eta\eta_3'^j\eta'_2\eta_1'^i$ is over word $u_1^k w u_2^k$, and goes from p to p'. As T is trimmed, by Lemma 1.b, this gives arbitrarily many accepting paths over a same word. □

A.4 Proof of Lemma 4

Proof. We first define the following set of positions in u: $P = \{i \in \mathbb{N}_{>0} \mid x \leq i \leq y \wedge u_{x,i} \in \Sigma^*_{\mathsf{wn}}\}$. Then for each $i \in P$, we define $r_i = s^u_{x,i}$ and $X_i = \{q \in$

$Q \mid \mathsf{induced}^u_{x,i}(p,q) > 0$ for some $p\}$. Intuitively, r_i corresponds to the multiplicity associated with the well-nested subword $u_{x,i}$ and X_i is the set of states that can be reached after this subword (along an accepting path over u). Note that we have $x \in P$, as we have $u_{x,x} = \epsilon \in \Sigma^*_{\mathsf{wn}}$, and X_x is the set of states that accepting paths of T on u can go through at position x. In particular, this entails $r_x = s^u_{x,x} \leq n$. For all $i, j \in P$, such that $i < j$, we have:

1. $\forall p \in X_i, \exists q \in X_j$ such that there is a path $p \xrightarrow{u_{i+1,j}} q$ in T
2. $\forall q \in X_j, \exists p \in X_i$ such that there is a path $p \xrightarrow{u_{j,i+1}} q$ in T
3. $r_i \leq r_j$
4. $r_j \leq r_i \times s^u_{i,j}$

Properties (3) and (4) follow Lemma 3. Let C be the set defined as $C = \{i \in P \mid \forall j \in P, j < i \Rightarrow r_j < r_i\}$. We note the elements of C in ascending order as $i_1, i_2, \cdots, i_c$, with $c = |C|$. Then we have $r_{i_j} < r_{i_{j'}}$ for any $1 \leq j < j' \leq c$.

We will now prove that there exists $1 \leq j < c$ such that $s^u_{i_j,i_{j+1}} \geq Nl$. By contradiction, we suppose that this property does not hold, *i.e.*:

$$\forall 1 \leq j < c, s^u_{i_j,i_{j+1}} < Nl \qquad (\dagger)$$

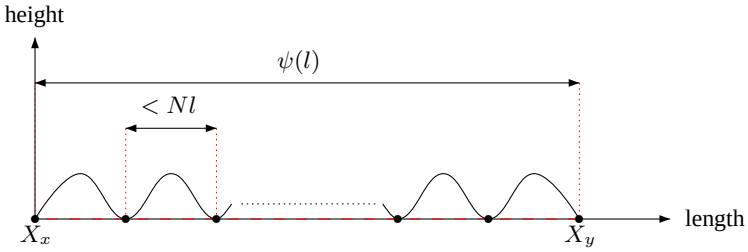


Fig. 6. $s^u_{i_j,i_{j+1}} < Nl$

Property (†) combined with property (4) implies $r_{i_{j+1}} < r_{i_j} \times Nl$ for any $1 \leq j < c$. We thus obtain $r_{i_c} < r_{i_1} \times (Nl)^{c-1}$. By definition, we have $r_{i_1} = r_x \leq n$, and $r_{i_c} = s^u_{x,y} \geq \psi(l) = n(Nl)^{2^n}$. We thus obtain $c - 1 > 2^n$, which entails that there exist two indices $j \neq j'$ such that $X_{i_j} = X_{i_{j'}}$. We note $X = X_{i_j}$. Let $v = u_{i_j,i_{j'}}$. By construction $v \in \Sigma^*_{\mathsf{wn}}$. Now we construct the multigraph $\mathcal{X}_v = (X, E_v)$ with $E_v \subseteq X \times X \times \mathbb{N}$ defined as follows: $\forall p, q \in Q, m \in \mathbb{N}$, for each path $\eta = p \xrightarrow{v|m} q$ such that $\eta'\eta\eta''$ is an accepting path on u for some path η' on word $u_{0,i}$, and η'' on word $u_{j,|u|}$, we construct an edge $p \xrightarrow{m} q \in E_v$. Note in particular that if there are two different paths on word v with the same multiplicity m going from state p to state q, then the edge $p \xrightarrow{m} q$ occurs twice in the multiset E_v.

Because of (1) and (2) and the fact that $X_{i_j} = X_{i_{j'}}$, each vertex from X has both an in-degree and an out-degree greater or equal than 1. Suppose for the sake of contradiction that the two following properties hold simultaneously:

(a) each arc of E_v has a multiplicity equal to 1,
(b) each node of $\mathcal{X}_v$ has an in-degree and an out-degree equal to 1.

In this case, one can observe that $\mathcal{X}_v$ is a simple graph (not a multigraph), and then that $\mathsf{induced}^u_{i_j,i_{j'}}$ is exactly the incidence matrix of this graph. Moreover, the graph is functional, and thus this matrix is a permutation matrix M. As a consequence of Lemma 3, we obtain $\mathsf{induced}^u_{x,i_j} = \mathsf{induced}^u_{x,i_{j'}} \times M$ and then $r_{i_j} = r_{i_{j'}}$, which is a contradiction.

We now distinguish two cases whether assertion (a) or assertion (b) does not hold. We first suppose that (b) holds and (a) does not hold. Consider the arc of E_v that has a multiplicity $m > 1$. Thanks to property (b), this arc belongs to a cycle in graph $\mathcal{X}_v$. This cycle then corresponds to a "heavy cycle" in T, in the sense of pattern **(S1.a)**.

Consider now that the assertion (b) is false. W.l.o.g., this entails that there exists a vertex $p \in X$ whose out-degree is at least two. By definition, there are two different paths η, η' in T of the form:

$$\eta : p \xrightarrow{v} q \text{ and } \eta' : p \xrightarrow{v} q'$$

Note that the paths are distinct but q and q' could be equal. Since each vertex of $\mathcal{X}$ has at least one successor, q and q' allow to reach a cycle along the word v^l for some $l \in \mathbb{N}_{>0}$. Moreover, since each vertex of $\mathcal{X}$ has at least one predecessor, p is accessible from a cycle. The overall situation is depicted on Figure 7, where φ_i denote the state around which are the cycles.

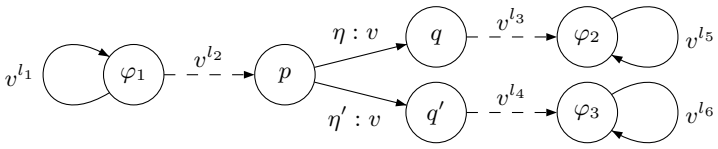


Fig. 7. Finding the dumbbell using graph $\mathcal{X}_v$.

If $\varphi_1 \neq \varphi_2$, then the pattern **(S1.b)** can be exhibited in T using states φ_1 to φ_2 and a well-chosen iteration of the two cycles (in such a way that the powers of word v are matching). The situation is similar if $\varphi_1 \neq \varphi_3$. If $\varphi_1 = \varphi_2 = \varphi_3$, then pattern **(S1.b)** is present in T using the two different paths η and η'. There are two different cycles around state p, one using η going through locations q, $\varphi_2 = \varphi_1$ and p, and another one using η' going through locations q', $\varphi_3 = \varphi_1$ and p. These two cycles are different (as $\eta \neq \eta'$), and can be iterated so that they are on the same word v^l for some $l > 0$. This yields the expected dumbbell.

Finally, we have proven that if Property (†) holds, then T contains pattern **(S1)**, which contradicts our hypothesis. Thus there exists $1 \leq j < c$ such that $s^u_{i_j,i_{j+1}} \geq Nl$. Let two positions x'' and y'' defined by $y'' = i_{j+1}$ and $x'' = \max\{i \in P \mid i < y''\}$. By definition of C, we have $r_{i_j} = r_{x''}$. This means that $s^u_{x,i_j} = s^u_{x,x''}$, *i.e.* the sums of multiplicities of accepting paths over u between positions x and i_j on one side, and between x and x'' on the other side, are equal. This entails that when considering the same sums between positions larger than i_j, the equality will also hold. In other terms, we can deduce $s^u_{i_j,i_{j+1}} = s^u_{x'',i_{j+1}}$. Moreover, by our choice of x'' and y'', there is no position $z \in P$ such that $x'' < z < y''$. Two cases are possible, either $u_{x'',y''} \in \Sigma_\iota$, or $u_{x'',y''} = cwr$, with $c \in \Sigma_c$, $r \in \Sigma_r$ and $w \in \Sigma^*_{\mathsf{wn}}$. The property $s^u_{x'',y''} \geq Nl$

excludes the first case. We can thus consider positions $x' = x'' + 1$ and $y' = y'' - 1$ which fulfill the conditions $x \leq x' \leq y' \leq y$, $u_{x',y'} \in \Sigma^*_{\mathsf{wn}}$ and $h_u(x') = h_u(x) + 1$. Finally, $s^u_{x'',y''} \geq Nl = (n^2L)^2|\Gamma|l$ implies that $s^u_{x',y'} \geq l$ as expected. □

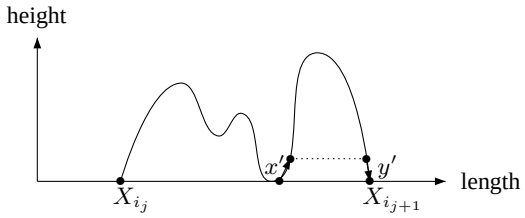


Fig. 8. $s^u_{x'',y''} \geq Ll$

A.5 Proof of Lemma 5

Proof. To prove this result, we assume that T is infinite but does not comply with **(S1)**, and prove it complies with **(S2)**.

Let a word $u \in \mathcal{L}(T)$ such that $\langle u \rangle \geq \psi^H(1)$ where $H = 2^{n^2}$, and let $k = |u|$. First, we define a partial function $\Phi_u : \mathbb{N} \times \mathbb{N} \to \mathbb{N} \times \mathbb{N}$. The domain of Φ_u is the set of couple of positions (x, y) such that $0 \leq x \leq y \leq k$, $u_{x,y} \in \Sigma^*_{\mathsf{wn}}$ and $s^u_{x,y} \geq \psi(1)$.Let (x, y) be a couple of positions in $\mathsf{Dom}(\Phi_u)$ and $l_{\max}$ be the largest $l \in \mathbb{N}_{>0}$ such that $s^u_{x,y} \geq \psi(l_{\max})$, Then by Lemma 4 there exist two positions x' and y' such that $x < x' \leq y' < y$, $u_{x',y'} \in \Sigma^*_{\mathsf{wn}}$, $h_u(x') = h_u(x)+1$ and $s^u_{x',y'} \geq l_{\max}$. We pick (x', y') minimal in lexicographical order, and then define $\Phi_u(x, y) = (x', y')$. We consider the (finite) sequence $(\chi_i)_i \in (\mathbb{N} \times \mathbb{N})^{\mathbb{N}}$ defined by $\chi_0 = (0, k)$ and, for $i \geq 1$, χ_i is defined iff $\chi_{i-1} \in \mathsf{Dom}(\Phi_u)$, and then defined as $\chi_i = \Phi_u(\chi_{i-1})$.

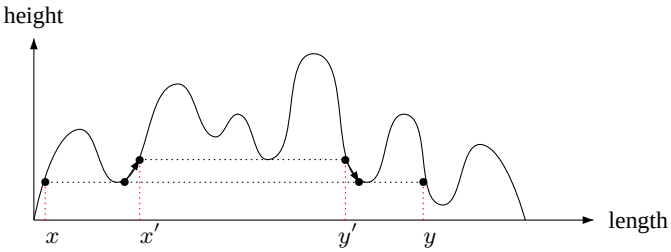


Fig. 9. $s^u_{x,y} \geq \psi(l_{\max})$

By definition of mapping Φ_u, we have for any $i \geq 1$, $h_u(\chi_i) = h_u(\chi_{i-1})+1$. Since h_u is finite this entails that sequence $(\chi_i)_i$ is finite, and we represent it as $(\chi_i)_{0 \leq i \leq L}$. Note that the sequence stops iff the last term does not belong to $\mathsf{Dom}(\Phi_u)$, which means that $s^u_{\chi_L} < \psi(1)$ (the other conditions are fulfilled). For each index $0 \leq i \leq L$ we define a value $r_i \in \mathbb{N}_{>0}$ and a set of pairs of states $X_i \subseteq Q \times Q$ as follows: (we let $\chi_i = (x, y)$)

- $r_i = s^u_{\chi_i}$
- $X_i = \{(p_1, p_2) \in Q \times Q \mid$ there exist a path $\eta = p_1 \xrightarrow{u_{x,y}} p_2$, a path η' on word $u_{0,x}$ and a path η'' on word $u_{y,k}$ such that $\eta'\eta\eta''$ is an accepting path of $T\}$

Note that by definition of $s^u_{\chi_i}$, we have $r_{i-1} \geq r_i$ for any $1 \leq i \leq L$. Let C be the set defined as $C = \{i \in [0, L] \cap \mathbb{N} \mid \forall j \in \mathbb{N}, j < i \Rightarrow r_j > r_i\}$. We note the elements of C in ascending order as $i_0, i_1, \ldots, i_c$ with $c = |C| - 1$. Note that $r_{i_j} > r_{i_{j'}}$, for $0 \leq j < j' \leq c$. We have $r_{i_0} = r_0 = s^u_{\chi_0} = \langle u \rangle \geq \psi^H(1)$. By Lemma 4 this implies $r_{i_h} \geq \psi^{H-h}(1)$ for $0 \leq h \leq \min\{H, c\}$. As we have $r_{i_c} = r_L = s^u_{\chi_L} < \psi(1)$, this entails $c \geq H$. As there are $c+1$ elements in C, this implies that there exist two distinct indices j and j' such that $X_{i_j} = X_{i_{j'}}$.

The rest of the proof follows the same lines as proof of Lemma 4, but to identify pattern **(S2)**, we consider a graph where vertices are pairs of states. We let $X = X_{i_j}$, $(x, y) = \chi_{i_j}$, $(x', y') = \chi_{i_{j'}}$, $u_1 = u_{x,x'}$ be the "call loop" word, $u_2 = u_{y',y}$ be the "return loop" word, and $w = u_{x',y'}$ be the well-nested word between the "call loop" and the "return loop" (see pattern **(S2)**). Now we construct the multigraph $\mathcal{X} = (X, E)$ where $E \subseteq X \times X \times \mathbb{N}_{>0} \times \mathbb{N}_{>0}$ is defined as follows: for each pair of paths $\eta_1 : p_1 \xrightarrow{v_1|m_1} q_1, q_2 \xrightarrow{v_2|m_2} p_2$ such that $\eta' \eta_1 \eta \eta_2 \eta''$ is an accepting path on u, for some paths η', η and η'' on words u_{0,x_1}, w and $u_{x_2,k}$ respectively, we add the edge $(p_1, q_1) \xrightarrow{(m_1,m_2)} (p_2, q_2)$ in E. Note that $\mathcal{X}$ is a multigraph, thus the same edge can occur more than once in E.

Note that a path in $\mathcal{X}$ corresponds to a path in T in the following sense: let $(p_0, q_0) \xrightarrow{(m^0_1,m^0_2)} (p_1, q_1) \cdots (p_{\mu-1}, q_{\mu-1}) \xrightarrow{(m^\mu_1,m^\mu_2)} (p_\mu, q_\mu)$ be a path of $\mathcal{X}$, then $p_0 \xrightarrow{u_1|m^0_1} p_1 \cdots p_{\mu-1} \xrightarrow{u_1|m^\mu_1} p_\mu \xrightarrow{w|m} q_\mu \xrightarrow{u_2|m^\mu_2} q_{\mu-1} \cdots q_1 \xrightarrow{u_2|m^0_1} q_0$ is a path of T that can be extended to an accepting path, for some $m \in \mathbb{N}_{>0}$.

Note that because $X_{i_j} = X_{i_{j'}}$, each vertex from X has both an in-degree and an out-degree greater or equal than 1. Suppose for the sake of contradiction that the two following properties hold simultaneously:

(a) each arc of E has a multiplicity equal to $(1, 1)$,
(b) each node of $\mathcal{X}$ has an in-degree and an out-degree equal to 1.

In this case, one can observe that $\mathcal{X}$ is a simple graph (not a multigraph), and then that $r_{i_j} = r_{i_{j'}}$, which is a contradiction.

We now distinguish two cases whether assertion (a) or assertion (b) does not hold. We first suppose that (b) holds and (a) does not hold. Consider the arc of E_v that has a multiplicity $m > 1$. Thanks to property (b), this arc belongs to a cycle in graph $\mathcal{X}$. One can verify that this cycle corresponds to a "heavy cycle" in T, in the sense of pattern **(S2.a)**.

Now we consider that assertion (b) does not hold. W.l.o.g., this entails that there exists $(p, q) \in X$ with at least two successors: there are two distinct edges

$$(p, q) \xrightarrow{(m^1_1,m^1_2)} (p_1, q_1) \text{ and } (p, q) \xrightarrow{(m^2_1,m^2_2)} (p_2, q_2)$$

We let $\varphi_1 = (p_1, q_1)$, $\varphi_2 = (p_2, q_2)$ and consider the four paths of T: $\eta_1 = p \xrightarrow{u_1|m^1_1} p_1$, $\eta_2 = p \xrightarrow{u_1|m^2_1} p_2$, $\eta'_1 = q \xrightarrow{u_2|u^1_2} q_1$ and $\eta'_2 = q \xrightarrow{u_2|u^2_2} q_2$. By construction φ_1 and φ_2 can be equal but we have $(\eta_1, \eta'_1) \neq (\eta_2, \eta'_2)$.

Since each vertex of $\mathcal{X}$ has at least one successor, φ_1 and φ_2 allow to reach cycles in $\mathcal{X}$. Moreover, since each vertex of $\mathcal{X}$ has at least one predecessor, (p, q) is reachable from a cycle in $\mathcal{X}$. The overall situation is depicted on Figure 10.

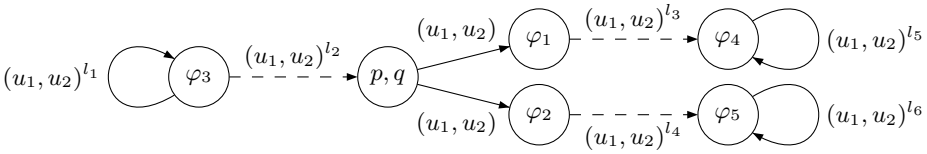


Fig. 10. Finding the dumbbell using graph $\mathcal{X}$.

We let $\varphi_3 = (p_3, q_3)$, $\varphi_4 = (p_4, q_4)$ and $\varphi_5 = (p_5, q_5)$ and distinguish three cases. If $\varphi_3 \neq \varphi_4$, then we can identify patter **(S2.b)** using the path in $\mathcal{X}$ going from φ_3 to φ_4 (the construction is similar to that done on $\mathcal{X}_v$ in the proof of Lemma 4). The same reasoning holds if $\varphi_3 \neq \varphi_5$. If $\varphi_3 = \varphi_4 = \varphi_5$, then pattern **(S2.b)** can be exhibited using the two different edges outgoing from (p, q). The constructions are similar to that done on $\mathcal{X}_v$ in the proof of Lemma 4. This concludes the proof. □

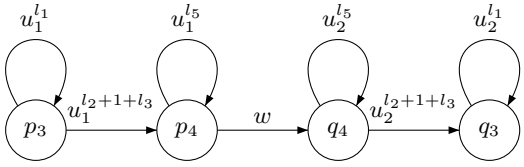


Fig. 11. $\varphi_3 \neq \varphi_4$

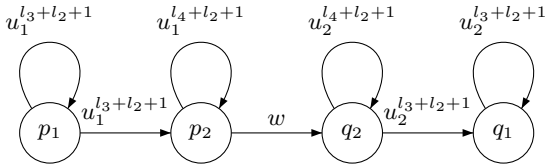


Fig. 12. $\varphi_3 = \varphi_4 = \varphi_5$

A.6 Proof of Proposition 1

Proof. We proceed successively with $\mathcal{S}_0$, $\mathcal{S}_1$, $\mathcal{S}_2$ and $\mathcal{S}_3$.

We prove that for all couple $c = (p, q)$, $c \in \mathcal{S}_0$ if and only if the following property holds:

$$\exists u \in \Sigma^*_{\mathsf{wn}} \text{ and a path } \eta : p \xrightarrow{u} q \text{ of } T \tag{1}$$

First the forward direction. We proceed by induction. We show that any couple in $\mathcal{S}_0$ satisfies (1). A couple can be added to $\mathcal{S}_0$ in four different ways:

rule 1.1 Let (p, p) be a couple added to $\mathcal{S}_0$ by rule 1.1. So $p \in Q$. Observe that η_ϵ is a path of T. Then (p, p) satisfies (1).

rule 1.2 Let (p, q) be a couple added to $\mathcal{S}_0$ by rule 1.2. So there exists $t = (p, a, q) \in \delta_\iota$. Observe that $a \in \Sigma^*_{\mathsf{wn}}$ and $p \xrightarrow{a} q$ is a path of T because of t. Then (p, q) satisfies (1).

rule 1.3 Let (p, q') be a couple added to $\mathcal{S}_0$ by rule 1.3. So there exist $(p, q), (q, q') \in \mathcal{S}_0$. By induction there exist two words $u, u' \in \Sigma^*_{\mathsf{wn}}$ such that $p \xrightarrow{u} q$ and $q \xrightarrow{u'} q'$ are paths of T. Observe that $uu' \in \Sigma^*_{\mathsf{wn}}$ and $p \xrightarrow{uu'} q'$ is a path of T. Then (p, q') satisfies (1).

rule 1.4 Let (p', q') be a couple added to $\mathcal{S}_0$ by rule 1.4. So there exist $(p, q) \in \mathcal{S}_0$, $(p', c, \gamma, p) \in \delta_c$ and $(q, r, \gamma, q') \in \delta_r$. By induction there exists a word $u \in \Sigma^*_{\mathsf{wn}}$ such that $p \xrightarrow{u} q$ is a path of T. Observe that $cur \in \Sigma^*_{\mathsf{wn}}$, $p' \xrightarrow{cur} q'$ is a path of T. Then (p, q) satisfies (1).

Then the backward direction. By contradiction suppose that there exist $u \in \Sigma^*_{\mathsf{wn}}$ and two states $p, q \in Q$ such that $p \xrightarrow{u} q$ is a path of T but $(p, q) \notin \mathcal{S}_0$. We choose u such that $|u|$ is minimal. If $|u| \leq 1$, then by rule 1.1 and 1.2, $(p, q) \in \mathcal{S}_0$, contradiction. Otherwise we can decompose u in two ways: either $u = u_1 u_2$ such that $u_1, u_2 \in \Sigma^*_{\mathsf{wn}} \setminus \{\epsilon\}$, or $u = cu'r$ such that $c \in \Sigma_c$, $r \in \Sigma_r$ and $u' \in \Sigma^*_{\mathsf{wn}}$. We consider the first case. $u_1 \neq \epsilon$ and $u_2 \neq \epsilon$, so we can decompose $p \xrightarrow{u} q$ as $p \xrightarrow{u_1} p' \xrightarrow{u_2} q$. Since u is minimal we have $|u| > |u_1|$ and $|u| > |u_2|$, this entails $(p, p') \in \mathcal{S}_0$ and $(p', q) \in \mathcal{S}_0$. Then by rule 1.3, $(p, q) \in \mathcal{S}_0$, contradiction. Now we consider the second case. We can decompose $p \xrightarrow{u} q$ as $p \xrightarrow{c} p' \xrightarrow{u'} q' \xrightarrow{r} q$. Since u is minimal and $|u| > |u'|$, $(p', q') \in \mathcal{S}_0$. Then by rule 1.4, $(p, q) \in \mathcal{S}_0$, contradiction.

We prove that for all tuple $c = (p_1, q_1, p_2, q_2, p_2, q_3, B)$, $c \in \mathcal{S}_1$ if and only if the following property holds:

$$\begin{aligned} &\exists u \in \Sigma^*_{\mathsf{wn}} \text{ and three paths } \eta_i : p_i \xrightarrow{u} q_i \text{ of } T \text{ for } i \in \{1, 2, 3\} \\ &\qquad\qquad \text{such that } B = (\eta_1 \neq \eta_2 \neq \eta_\epsilon \vee \langle \eta_1 \rangle > 1) \end{aligned} \tag{2}$$

First the forward direction. We show that every rule preserves (2). We proceed by induction. We show that any tuple in $\mathcal{S}_1$ satisfies (2). A tuple can be added to $\mathcal{S}_1$ in four different ways:

rule 2.1 Let $(p_1, p_1, p_2, p_2, p_3, p_3, B)$ be a tuple added to $\mathcal{S}_1$ by rule 2.1. So there exist $p_i \in Q$ for $i \in \{1, 2, 3\}$. Observe that η_ϵ is a path of T. Then $(p_1, p_1, p_2, p_2, p_3, p_3, B)$ satisfies (2) with $B = \bot$.

rule 2.2 Let $(p_1, p'_1, p_2, p'_2, p_3, p'_3, B)$ be a tuple added to $\mathcal{S}_1$ by rule 2.2. So there exist $t_i = (p_i, a, q_i) \in \delta_\iota$ for $i \in \{1, 2, 3\}$ such that $B = \lambda(t_1) > 1 \vee (t_1 \neq t_2)$. Observe that $a \in \Sigma^*_{\mathsf{wn}}$ and $\eta_i = p_i \xrightarrow{a} q_i$ is a path of T because of t_i. Then we have $\langle \eta_1 \rangle = \lambda(t_1)$ and $\eta_1 \neq \eta_2$ iff $t_1 \neq t_2$. Then $(p_1, q_1, p_2, q_2, p_3, q_3, B)$ satisfies (2).

rule 2.3 Let $(p_1, q'_1, p_2, q'_2, p_3, q'_3, B'')$ be a tuple added to $\mathcal{S}_1$ by rule 2.3. So there exist $(p_1, q_1, p_2, q_2, p_3, q_3, B), (q_1, q'_1, q_2, q'_2, q_3, q'_3, B') \in \mathcal{S}_1$ such that $B'' = B \vee B'$. By induction there exist two words $u, u' \in \Sigma^*_{\mathsf{wn}}$ such that $\eta_i = p_i \xrightarrow{u} q_i$ and $\eta'_i = q_i \xrightarrow{u'} q'_i$ are paths of T for $i \in \{1, 2, 3\}$, with $B = \langle \eta_1 \rangle > 1 \vee \eta_\epsilon \neq \eta_1 \neq \eta_2$ and $B' = \langle \eta'_1 \rangle > 1 \vee \eta_\epsilon \neq \eta'_1 \neq \eta'_2$. Observe that $uu' \in \Sigma^*_{\mathsf{wn}}$ and $\eta''_i = p_i \xrightarrow{uu'} q_i$

is a path of T. Moreover $\langle \eta_1'' \rangle = \langle \eta_1 \rangle \langle \eta_1' \rangle$ and thus $\langle \eta_1'' \rangle > 1$ iff $\langle \eta_1 \rangle > 1$ or $\langle \eta_1' \rangle > 1$. In addition, one also has $\eta_1'' \neq \eta_2'' \neq \eta_\epsilon$ iff $\eta_\epsilon \neq \eta_1 \neq \eta_2$ or $\eta_\epsilon \neq \eta_1' \neq \eta_2'$. Finally, $\langle \eta_1'' \rangle > 1 \vee \eta_1'' \neq \eta_2'' \neq \eta_\epsilon$ is logically equivalent to $B \vee B'$. Then $(p_1, q_1', p_2, q_2', p_3, q_3', B'')$ satisfies (2).

rule 2.4 Let $(p_1', q_1', p_2', q_2', p_3', q_3', B')$ be a tuple added to $\mathcal{S}_1$ by rule 2.3. So there exist $(p_1, q_1, p_2, q_2, p_3, q_3, B) \in \mathcal{S}_1$, $t_i = (p_i', c, \gamma, p_i) \in \delta_c$ and $t_i' = (q_i, r, \gamma, q_i') \in \delta_r$ for $i \in \{1, 2, 3\}$ such that $B' = B \vee (\lambda(t_1) > 1 \vee \lambda(t_1') > 1 \vee t_1 \neq t_2 \vee t_1' \neq t_2')$. By induction there exists a word $u \in \Sigma_{\mathsf{wn}}^*$ such that $\eta_i = p_i \xrightarrow{u} q_i$ is a path of T with $B = \langle \eta_1 \rangle > 1 \vee \eta_\epsilon \neq \eta_1 \neq \eta_2$. Observe that $cur \in \Sigma_{\mathsf{wn}}^*$, $\eta_i' = p_i' \xrightarrow{cur} q_i'$ is a path of T and $\langle \eta_1' \rangle > 1 \vee \eta_1' \neq \eta_2' \neq \eta_\epsilon$ is logically equivalent to $(B \vee \lambda(t_1) > 1 \vee \lambda(t_1') > 1 \vee t_1 \neq t_2 \vee t_1' \neq t_2')$. Then $(p_1, q_1, p_2, q_2, p_3, q_3, B')$ satisfies (2).

Then the backward direction. By contradiction suppose that there exist $u \in \Sigma_{\mathsf{wn}}^*$ and states $p_i, q_i \in Q$ such that $\eta_i = p_i \xrightarrow{u} q_i$ is a path of T for $i \in \{1, 2, 3\}$, but $(p_1, q_1, p_2, q_2, p_3, q_3, B) \notin \mathcal{S}_1$ with $B = \langle \eta_1 \rangle > 1 \vee \eta_1 \neq \eta_2 \neq \eta_\epsilon$. We choose u such that $|u|$ is minimal. If $|u| \leq 1$, then by rule 2.1 and 2.2, $(p_1, q_1, p_2, q_2, p_3, q_3, B) \in \mathcal{S}_1$, contradiction. Otherwise we can decompose u in two ways: either $u = u_1 u_2$ such that $u_1, u_2 \in \Sigma_{\mathsf{wn}}^* \setminus \{\epsilon\}$, or $u = cu'r$ such that $c \in \Sigma_c$, $r \in \Sigma_r$ and $u' \in \Sigma_{\mathsf{wn}}^*$. We consider the first case. $u_1 \neq \epsilon$ and $u_2 \neq \epsilon$, so we can decompose η_i as $\eta_i' = p_i \xrightarrow{u_1} p_i'$ and $\eta_i'' = p_i' \xrightarrow{u_2} q_i$ for $i \in \{1, 2, 3\}$. Since u is minimal, we have $|u| > |u_1|$ and $|u| > |u_2|$, this entails $(p_1, p_1', p_2, p_2', p_3, p_3', B') \in \mathcal{S}_1$ with $B' = \langle \eta_1' \rangle > 1 \vee \eta_1' \neq \eta_2' \neq \eta_\epsilon$ and $(p_1', q_1, p_2', q_2, p_3', q_3, B'') \in \mathcal{S}_1$ with $B'' = \langle \eta_1'' \rangle > 1 \vee \eta_1'' \neq \eta_2'' \neq \eta_\epsilon$. By rule 1.3, $(p_1, q_1, p_2, q_2, p_3, q_3, B' \vee B'') \in \mathcal{S}_1$. Observe that $\langle \eta_1 \rangle > 1 \vee \eta_1 \neq \eta_2 \neq \eta_\epsilon$ is logically equivalent to $B' \vee B''$. Contradiction. Now we consider the second case. We can decompose η_i as $\eta_i = \eta_i' \eta_i'' \eta_i'''$ where $\eta_i' = p_i \xrightarrow{c} p_i'$, $\eta_i'' = p_i' \xrightarrow{u'} q_i'$ and $\eta_i''' = q_i' \xrightarrow{r} q_i$ for $i \in \{1, 2, 3\}$. Since u is minimal and $|u| > |u'|$, $(p_1', q_1', p_2', q_2', p_3', q_3', B') \in \mathcal{S}_1$ with $B' = \langle \eta_1'' \rangle > 1 \vee \eta_1'' \neq \eta_2'' \neq \eta_\epsilon$. Then by rule 1.4, $(p_1, q_1, p_2, q_2, p_3, q_3, B'') \in \mathcal{S}_1$ with $B'' = (B' \vee \langle \eta_1' \rangle > 1 \vee \langle \eta_1''' \rangle > 1 \vee \eta_1' \neq \eta_2' \vee \eta_1''' \neq \eta_2''')$, Observe that $\langle \eta_1 \rangle > 1 \vee \eta_1 \neq \eta_2$ is logically equivalent to B'', contradiction.

For the end of the proof we define the following notation: let u, u' be two words such that $uu' \in \Sigma_{\mathsf{wn}}^*$ and $\eta_1 : p \xrightarrow{u} q$, $\eta_2 : q' \xrightarrow{u'} p'$ be two paths of T. As $uu' \in \Sigma_{\mathsf{wn}}^*$, there exist $(p, \bot) \xrightarrow{u} (q, \sigma)$ and $(q', \sigma') \xrightarrow{u'} (p', \bot)$ two runs of T for some $\sigma, \sigma' \in \Gamma^*$. Then we write that η_1 and η_2 are *matching paths* iff $\sigma = \sigma'$.

We prove that for all tuple $c = (p_1, q_1, p_2, q_2, p_3, q_3, q_3', p_3', q_2', p_2', q_1', p_1', B)$, $c \in \mathcal{S}_2$ if and only if the following property holds:

$$\begin{aligned} \exists uu' \in \Sigma_{\mathsf{wn}}^* \text{ and paths } \eta_i : p_i \xrightarrow{u} q_i \text{ and } \eta_i' : q_i' \xrightarrow{u'} p_i' \text{ of } T \text{ for } i \in \{1, 2, 3\} \\ \text{such that } \eta_i \text{ and } \eta_i' \text{ are matching paths of } T \text{ for } i \in \{1, 2, 3\} \\ \text{and } B = (\eta_1 \neq \eta_2 \neq \eta_\epsilon \vee \eta_1' \neq \eta_2' \neq \eta_\epsilon \vee \langle \eta_1 \rangle > 1 \vee \langle \eta_1' \rangle > 1) \end{aligned} \tag{3}$$

First the forward direction. We show that every rule preserves (3). We proceed by induction. We show that any tuple in $\mathcal{S}_2$ satisfies (3). A tuple can be added to $\mathcal{S}_2$ in three different ways:

rule 3.1 Let $(p_1, q_1, p_2, q_2, p_3, q_3, q'_3, p'_3, q'_2, p'_2, q'_1, p'_1, B'')$ be a tuple added to $\mathcal{S}_2$ by rule 3.1. So there exist $(p_1, q_1, p_2, q_2, p_3, q_3, B)$, $(q'_1, p'_1, q'_2, p'_2, q'_3, p'_3, B') \in \mathcal{S}_1$ such that $B'' = B \vee B'$. Then by (2) there exist $u, u' \in \Sigma^*_{\mathsf{wn}}$ and paths $\eta_i = p_i \xrightarrow{u} q_i$ and $\eta'_i = q'_i \xrightarrow{p_i{}'} p'_i$ of T for $i \in \{1, 2, 3\}$ with $B = \langle \eta_1 \rangle > 1 \vee \eta_1 \neq \eta_2$ and $B' = \langle \eta'_1 \rangle > 1 \vee \eta'_1 \neq \eta'_2$. Observe that $uu' \in \Sigma^*_{\mathsf{wn}}$ and η_i and η'_i for any $i \in \{1, 2, 3\}$ are trivially matching paths as u and u' are well-nested. Then $(p_1, q_1, p_2, q_2, p_3, q_3, q'_3, p'_3, q'_2, p'_2, q'_1, p'_1, B'')$ satisfies (3).

rule 3.2 Let $(p'_1, p_1, p'_2, p_2, p'_3, p_3, q_3, q'_3, q_2, q'_2, q_1, q'_1, B)$ be a tuple added to $\mathcal{S}_2$ by rule 3.2. So there exist $t_i = (p'_i, c, \gamma, p_i) \in \delta_c$ and $t'_i = (q_i, r, \gamma, q'_i) \in \delta_r$ for $i \in \{1, 2, 3\}$ such that $B = (\lambda(t_1) > 1 \vee \lambda(t'_1) > 1 \vee t_1 \neq t_2 \vee t'_1 \neq t'_2)$. We can consider paths $\eta_i = p'_i \xrightarrow{c} p_i$ and $\eta'_i = q_i \xrightarrow{c} q'_i$ for $i \in \{1, 2, 3\}$. Observe that $cr \in \Sigma^*_{\mathsf{wn}}$, η_i and η'_i are matching paths of T and $(\eta_1 \neq \eta_2 \vee \eta'_1 \neq \eta'_2 \vee \langle \eta_1 \rangle > 1 \vee \langle \eta'_1 \rangle > 1)$ is logically equivalent to B. Then $(p'_1, p_1, p'_2, p_2, p'_3, p_3, q_3, q'_3, q_2, q'_2, q_1, q'_1, B)$ satisfies (3).

rule 3.3 Let $(p''_1, p'_1, p''_2, p'_2, p''_3, p'_3, q'_3, q''_3, q'_2, q''_2, q'_1, q''_1, B'')$ be a tuple added to $\mathcal{S}_2$ by rule 3.2. So there exist $(p''_1, p_1, p''_2, p_2, p''_3, p_3, q_3, q''_3, q_2, q''_2, q_1, q''_1, B)$, $(p_1, p'_1, p_2, p'_2, p_3, p'_3, q'_3, q_3, q'_2, q_2, q'_1, q_1, B') \in \mathcal{S}_2$ such that $B'' = B \vee B'$. By induction there exist four words u, u', $\tilde{u}'$ and $\tilde{u}$ such that $u\tilde{u} \in \Sigma^*_{\mathsf{wn}}$, $u'\tilde{u}' \in \Sigma^*_{\mathsf{wn}}$, matching paths $\eta_i = p''_i \xrightarrow{u} p_i$ and $\tilde{\eta}_i = q_i \xrightarrow{\tilde{u}} q''_i$ and matching paths $\eta'_i = p_i \xrightarrow{u'} p'_i$ and $\tilde{\eta}'_i = q'_i \xrightarrow{\tilde{u}'} q_i$ for $i \in \{1, 2, 3\}$, with $B = \eta_1 \neq \eta_2 \vee \tilde{\eta}_1 \neq \tilde{\eta}_2 \vee \langle \eta_1 \rangle > 1 \vee \langle \tilde{\eta}_1 \rangle > 1$ and $B' = \eta'_1 \neq \eta'_2 \vee \tilde{\eta}'_1 \neq \tilde{\eta}'_2 \vee \langle \eta'_1 \rangle > 1 \vee \langle \tilde{\eta}'_1 \rangle > 1$. Observe that $uu'\tilde{u}'\tilde{u} \in \Sigma^*_{\mathsf{wn}}$, $\eta''_i = p''_i \xrightarrow{uu'} p'_i$ and $\tilde{\eta}''_i = q'_i \xrightarrow{\tilde{u}'\tilde{u}} q''_i$ are matching paths of T and $(\eta''_1 \neq \eta''_2 \vee \tilde{\eta}''_1 \neq \tilde{\eta}''_2 \vee \langle \eta''_1 \rangle > 1 \vee \langle \tilde{\eta}''_1 \rangle > 1)$ is logically equivalent to $B \vee B' = B''$. Then $(p''_1, p'_1, p''_2, p'_2, p''_3, p'_3, q'_3, q''_3, q'_2, q''_2, q'_1, q''_1, B'')$ satisfies (3).

Then the backward direction. By contradiction suppose that there exist two words u, u' and $p_i, q_i, q'_i, p'_i \in Q$ such that $uu' \in \Sigma^*_{\mathsf{wn}}$, $\eta_i = p_i \xrightarrow{u} q_i$ and $\tilde{\eta}_i = q'_i \xrightarrow{u'} p'_i$ are matching paths of T for $i \in \{1, 2, 3\}$, but $(p_1, q_1, p_2, q_2, p_3, q_3, q'_3, p'_3, q'_2, p'_2, q'_1, p'_1, B) \notin \mathcal{S}_2$ with $B = \langle \eta_1 \rangle > 1 \vee \langle \tilde{\eta}_i \rangle > 1 \vee \eta_1 \neq \eta_2 \vee \tilde{\eta}_i \neq \tilde{\eta}_i$. We choose u and u' such that $h_u(|u|)$ (the number of pending calls of u) is minimal. There exists a unique decomposition of u (resp. u') as $u = w_1 c w_2$ (resp. as $u' = w'_2 r w'_1$) with $w_1, w'_1, w_2 w'_2 \in \Sigma^*_{\mathsf{wn}}$, $c \in \Sigma_c$ and $r \in \Sigma_r$. We can thus decompose path η_i as $\eta_i = \eta'_i \eta''_i \eta'''_i$ where $\eta'_i = p_i \xrightarrow{w_1} \tilde{p}_i$, $\eta''_i = \tilde{p}_i \xrightarrow{c} \tilde{q}_i$, $\eta'''_i = \tilde{q}_i \xrightarrow{w_2} q_i$ and similarly for path $\tilde{\eta}_i$ with paths $\tilde{\eta}'''_i = q'_i \xrightarrow{w'_2} \tilde{q}'_i$, $\tilde{\eta}''_i = \tilde{q}'_i \xrightarrow{r} \tilde{p}'_i$, $\tilde{\eta}'_i = \tilde{p}'_i \xrightarrow{w'_1} p'_i$. Since $w_1, w'_1 \in \Sigma^*_{\mathsf{wn}}$ we have $(p_1, \tilde{p}_1, p_2, \tilde{p}_2, p_3, \tilde{p}_3, B')$, $(\tilde{p}'_3, p'_3, \tilde{p}'_2, p'_2, \tilde{p}'_1, p'_1, \tilde{B}') \in \mathcal{S}_1$ with $B' = \langle \eta'_1 \rangle > 1 \vee \eta'_1 \neq \eta'_2$ and $\tilde{B}' = \langle \tilde{\eta}'_1 \rangle > 1 \vee \tilde{\eta}'_1 \neq \tilde{\eta}'_2$. Then by rule 3.1, we have $(p_1, \tilde{p}_1, p_2, \tilde{p}_2, p_3, \tilde{p}_3, \tilde{p}'_3, p'_3, \tilde{p}'_2, p'_2, \tilde{p}'_1, p'_1, B' \vee \tilde{B}') \in \mathcal{S}_2$. Following the decomposition $u = w_1 c w_2$, we have $h_u(|u|) > h_{w_2}|w_2|$. Since $h_u(|u|)$ is minimal, this entails $(\tilde{q}_1, q_1, \tilde{q}_2, q_2, \tilde{q}_3, q_3, q'_3, \tilde{q}'_3, q'_2, \tilde{q}'_2, q'_1, \tilde{q}'_1, B'') \in \mathcal{S}_2$ with $B'' = (\langle \eta'''_1 \rangle > 1 \vee \langle \tilde{\eta}'''_1 \rangle > 1 \vee \eta'''_1 \neq \eta'''_2 \vee \tilde{\eta}'''_1 \neq \tilde{\eta}'''_2)$. By rule 3.2 applied on matching paths η''_i and $\tilde{\eta}''_i$, we have $(\tilde{p}_1, \tilde{q}_1, \tilde{p}_2, \tilde{q}_2, \tilde{p}_3, \tilde{q}_3, \tilde{q}'_3, \tilde{p}'_3, \tilde{q}'_2, \tilde{p}'_2, \tilde{q}'_1, \tilde{p}'_1, B''')$ with $B''' = \langle \eta''_1 \rangle > 1 \vee \langle \tilde{\eta}''_1 \rangle > 1 \vee \eta''_1 \neq \eta''_2 \neq \tilde{\eta}''_1 \neq \tilde{\eta}''_2$. By applying twice the rule 3.3, we obtain $(p_1, q_1, p_2, q_2, p_3, q_3, q'_3, p'_3, q'_2, p'_2, q'_1, p'_1, B_c) \in \mathcal{S}_2$ with $B_c = B' \vee \tilde{B}' \vee B'' \vee B'''$. We can verify that B is logically equivalent with B_c, contradiction.

We now prove that $\top \in \mathcal{S}_3$ iff T is infinite. First the forward direction. There are two possibilities of the presence of $\top$ in $\mathcal{S}_3$:

rule 4.1 There exists $(p, p, p, q, q, q, \top) \in \mathcal{S}_1$ by rule 4.1. By property (2) of $\mathcal{S}_1$, there exist $u \in \Sigma^*_{\mathsf{wn}}$ and paths $\eta_1 = p \xrightarrow{u} p$, $\eta_2 = p \xrightarrow{u} q$ and $\eta_3 = q \xrightarrow{u} q$ of T with $\langle \eta_1 \rangle > 1$ or $\eta_1 \neq \eta_2$. This corresponds to the definition of pattern **S1**, then T is infinite.

rule 4.2 There exist $(q, q') \in \mathcal{S}_0$ and $(p, p, p, q, q, q, q', q', q', p', p', p', \top) \in \mathcal{S}_2$ by rule 4.2. By property (1) of $\mathcal{S}_0$ and property (3) of $\mathcal{S}_2$, there exist three words u, u', w, such that $uu', w \in \Sigma^*_{\mathsf{wn}}$, and paths $\eta = q \xrightarrow{w} q'$, $\eta_1 = p \xrightarrow{u} p$, $\eta'_1 = p' \xrightarrow{u'} p'$, $\eta_2 = p \xrightarrow{u} q$, $\eta'_2 = q' \xrightarrow{u'} p'$, $\eta_3 = q \xrightarrow{u} q$ and $\eta'_3 = q' \xrightarrow{u'} q'$ such that η_i and η'_i are matching paths of T and $\langle \eta_1 \rangle > 1$ or $\langle \eta'_1 \rangle > 1$ or $\eta_1 \neq \eta_2$ or $\eta'_1 \neq \eta'_2$. This corresponds to the definition of pattern **S2**, then T is infinite.

Then the backward direction. Since T is infinite, we are able to find pattern **S1** or pattern **S2** in T. If pattern **S1** is present in T, then there exist $u \in \Sigma^*_{\mathsf{wn}}$ and paths $\eta_1 = p \xrightarrow{u} p$, $\eta_2 = p \xrightarrow{u} q$ and $\eta_3 = q \xrightarrow{u} q$ of T with $\langle \eta_1 \rangle > 1$ or $\eta_1 \neq \eta_2$. By property (2) of $\mathcal{S}_1$, $(p, p, p, q, q, q, \top) \in \mathcal{S}_1$. Then by rule 4.1, $\top \in \mathcal{S}_3$. If pattern **S2** is present in T, then there exist three words u, u', w, such that $uu', w \in \Sigma^*_{\mathsf{wn}}$, and paths $\eta = q \xrightarrow{w} q'$, $\eta_1 = p \xrightarrow{u} p$, $\eta'_1 = p' \xrightarrow{u'} p'$, $\eta_2 = p \xrightarrow{u} q$, $\eta'_2 = q' \xrightarrow{u'} p'$, $\eta_3 = q \xrightarrow{u} q$ and $\eta'_3 = q' \xrightarrow{u'} q'$ such that η_i and η'_i are matching paths of T and with $\langle \eta_1 \rangle > 1$ or $\langle \eta'_1 \rangle > 1$ or $\eta_1 \neq \eta_2$ or $\eta'_1 \neq \eta'_2$. By property (1) of $\mathcal{S}_0$ and property (3) of $\mathcal{S}_2$, $(q, q') \in \mathcal{S}_0$ and $(p, p, p, q, q, q, q', q', q', p', p', p', \top) \in \mathcal{S}_2$. Then by rule 4.2, $\top \in \mathcal{S}_3$. □

B Complements to Section 5

Algorithm 2 Computation of the multiplicity of a finite $\mathbb{N}$-VPA

Require: A finite $\mathbb{N}$-VPA T

$\mathcal{M} \leftarrow \{M_a \mid a \in \Sigma_\iota\} \cup \{M_\epsilon\}$
$\mathcal{M}' \leftarrow \emptyset$
repeat
  $\mathcal{M} \leftarrow \mathcal{M} \cup \mathcal{M}'$
  $\mathcal{M}' \leftarrow \{M_1.M_2 \mid M_1, M_2 \in \mathcal{M}\} \cup$
    $\{\sum_{\gamma \in \Gamma} M_{c,\gamma}.M.M_{r,\gamma} \mid M \in \mathcal{M}, c \in \Sigma_c, r \in \Sigma_r, \gamma \in \Gamma\}$
until $\mathcal{M}' \cup \mathcal{M} = \mathcal{M}$
return $\max\{\langle M \rangle \mid M \in \mathcal{M}\}$

C From Tree Automata to VPA

C.1 Tree automata with multiplicities

Let $\Sigma = \Sigma_0 \cup \ldots \cup \Sigma_L$ be a ranked alphabet. For $a \in \Sigma$, the rank of a, denoted by $\mathsf{rk}(a)$, equals m iff $a \in \Sigma_m$. $\mathcal{T}_\Sigma$ denotes the free Σ-algebra of finite ordered Σ-labeled

trees, *i.e.* $\mathcal{T}_\Sigma$ is the smallest set $\mathcal{T}$ satisfying (i) $\Sigma_0 \subseteq \mathcal{T}$, and (ii), if $a \in \Sigma_m$ and $t_0, \ldots, t_{m-1} \in \mathcal{T}$, then $a(t_0, \ldots, t_{m-1}) \in \mathcal{T}$. Note that (i) can be viewed as a subcase of (ii) if we allow m to equal 0.

Let $t = a(t_0, \ldots, t_{m-1}) \in \mathcal{T}_\Sigma$ for some $a \in \Sigma_m$ with $m \geq 0$. The set of nodes of t, $S(t)$ is the subset of $\mathbb{N}^*$ defined by $S(t) = \{\epsilon\} \cup \bigcup_{j=0}^{m-1} j \cdot S(t_j)$. t defines a map $\alpha_t : S(t) \to \Sigma$ mapping the nodes of t to their labels. We have:

$$\alpha_t(r) = \begin{cases} a & \text{if } r = \epsilon \\ \alpha_{t_j}(r') & \text{if } r = j \cdot r' \end{cases}$$

Tree automata A finite tree automaton over Σ is a triple $A = (Q, I, \delta)$ where:

- Q is a finite set of states
- $I \subseteq Q$ is the set of initial states
- $\delta \subseteq \bigcup_{m=0}^{L} Q \times \Sigma_m \times Q^m$ is the set of transitions. Given a transition $\tau = (q, a, q_0 \ldots q_{m-1}) \in \delta$, we denote by $\mathsf{rk}(\tau)$ the value $\mathsf{rk}(a)$.

Let $t = a(t_0, \ldots, t_{m-1}) \in \mathcal{T}_\Sigma$ and $q \in Q$. A q-computation of A for t consists of a transition $(q, a, q_0 \ldots q_{m-1}) \in \delta$ for the root and q_j-computations of A for the subtrees t_j, $j \in \{0 \ldots m-1\}$. Formally, a computation φ of A for t can be viewed as a map $\varphi : S(t) \to \delta$ satisfying for any $r \in S(t)$, if $\alpha_t(r) = a \in \Sigma_m$, then $\varphi(r) = (q, a, q_0 \ldots q_{m-1})$ and for any $0 \leq j \leq m-1$, $\varphi(r \cdot j) = (q_j, a_j, q_0^j \ldots q_{m_j-1}^j)$ where $a_j = \alpha_t(r \cdot j)$ and $m_j = \mathsf{rk}(a_j)$. φ is a q-computation of A for t whenever $\varphi(\epsilon)$ is of the form $(q, a, q_0 \ldots q_{m-1})$. A q-computation is accepting iff $q \in I$. A tree t is accepted by A iff there is an accepting computation of A for t. The language of A is the set of trees accepted by A and is denoted by $\mathcal{L}(A)$.

Weighted tree automata A weighted tree automaton over the alphabet Σ and the semiring $(\mathbb{N}, +, \cdot)$ is a pair $T = (A, \lambda)$ where $A = (Q, I, \delta)$ is a tree automaton and $\lambda : \delta \to \mathbb{N}_{>0}$ is a mapping assigning a multiplicity to each transition of A. Notions of computations, accepted computations and languages are lifted from tree automata to weighted tree automata. Let $t \in \mathcal{L}(A)$. The multiplicity of a computation φ of A for t, denoted by $\langle\varphi\rangle$, is the product of the multiplicities composing it, *i.e.* $\langle\varphi\rangle = \Pi_{r \in S(t)} \lambda(\varphi(r))$. The multiplicity of $t \in \mathcal{L}(A)$, denoted by $\langle t \rangle$, is defined as $\langle t \rangle = \sum\{\langle\varphi\rangle \mid \varphi \text{ accepting computation for } t\}$.

C.2 From tree automata (with multiplicities) to VPA (with multiplicities)

From trees to well-nested words Let $\Sigma = \Sigma_0 \cup \ldots \cup \Sigma_L$ be a ranked alphabet. We defined the structured alphabet $\hat{\Sigma} = \Sigma_c \cup \Sigma_r$ as follows:

$$\Sigma_c = \{\langle a \mid a \in \Sigma\}$$
$$\Sigma_r = \{a\rangle \mid a \in \Sigma\}$$

The encoding of a tree $t \in \mathcal{T}_\Sigma$ is a well-nested word over $\hat{\Sigma}$, denoted by $\mathsf{enc}(t)$, and defined inductively as follows: if $t = a(t_0, \ldots, t_{m-1}) \in \mathcal{T}_\Sigma$, then $\mathsf{enc}(t) = \langle a\ \mathsf{enc}(t_0) \ldots \mathsf{enc}(t_{m-1})\ a\rangle$. One can easily verify that for any $t \in \mathcal{T}_\Sigma$, we have $\mathsf{enc}(t) \in \hat{\Sigma}^*_{\mathsf{wn}}$.

Automata translation Let $T = (A, \lambda)$ where $A = (Q, I, \delta)$ be a tree automaton with multiplicities over Σ. We define an $\mathbb{N}$-VPA $T' = (A', \lambda')$ such that: [2]

$$\mathcal{L}(T') = \{\mathsf{enc}(t) \mid t \in \mathcal{L}(T)\} \text{ and } \forall t \in \mathcal{L}(T), \langle t \rangle_T = \langle \mathsf{enc}(t) \rangle_{T'} \tag{4}$$

We first define the VPA $A' = (Q', \Gamma', \delta', Q'_{in}, Q'_f)$ over the alphabet $\hat{\Sigma}$ as follows:

- $Q' = \{(q, i) \mid q \in I, 0 \leq i \leq 1\} \cup \{(\tau, i) \mid \tau \in \delta, 0 \leq i \leq \mathsf{rk}(\tau)\}$
- $Q'_{in} = \{(q, 0) \mid q \in I\}$
- $Q'_f = \{(q, 1) \mid q \in I\}$
- $\Gamma' = Q' \times \Sigma$

We now define δ' by its restrictions δ'_c and δ'_r on call and return symbols respectively:

$s \xrightarrow{\langle a,(s,a)} s' \in \delta'_c$ iff one of the following cases holds:

1. $s = (q, 0)$ and $s' = (\tau', 0)$ where $q \in I$ and $\tau' = (q, a, q'_0 \dots q'_{m'-1}) \in \delta$
2. $s = (\tau, i)$ and $s' = (\tau', 0)$ where $\tau = (q, b, q_0 \dots q_{m-1}) \in \delta$, $i < \mathsf{rk}(\tau)$, and $\tau' = (q_i, a, q'_0 \dots q'_{m'-1}) \in \delta$

Intuitively, the state s gives the rule that is applied, and at which position in the rule we are. In the first case, this is an initialization rule, for some state q. We require that the new rule τ' starts from a root in state q. In the second case, we were applying rule τ, at position i. Thus the current state was q_i. We thus require that the new rule we apply (τ') starts from a root in state q_i.

$s \xrightarrow{a\rangle,(s'',a)} s' \in \delta'_r$ iff one of the following cases holds:

1. $s'' = (q, 0)$, $s' = (q, 1)$ and $s = (\tau', \mathsf{rk}(\tau'))$ where $q \in I$ and $\tau' = (q, a, q'_0 \dots q'_{m'-1}) \in \delta$
2. $s'' = (\tau, i)$, $s' = (\tau, i + 1)$ and $s = (\tau', \mathsf{rk}(\tau'))$ where $\tau = (q, b, q_0 \dots q_{m-1}) \in \delta$, $i < \mathsf{rk}(\tau)$, and $\tau' = (q_i, a, q'_0 \dots q'_{m'-1}) \in \delta$

To read a return symbol, we should have finished the rule we were applying. This is what is required with condition $s = (\tau', \mathsf{rk}(\tau'))$ in both cases. Then, we recover from the stack symbol s'' what is the rule we were applying on the root, and we move one position forward in this rule. For instance, in the second case, we go from (τ, i) to $(\tau, i + 1)$. We also check that the rule that is finished was on the good state, *i.e.* q in the first case, and q_i in the second case.

Last, we define the multiplicity mapping λ' as follows: for any transition $d \in \delta'_r$, we let $\lambda'(d) = 1$, and for any transition $d = s \xrightarrow{\langle a,(s,a)} s' \in \delta'_c$ where $s' = (\tau', 0)$, we let $\lambda'(d) = \lambda(\tau')$. Intuitively, a transition in the tree automaton is applied twice in the VPA, once when the call symbol is read, and once when the return symbol is read. We thus report its multiplicity only in the case of the call symbol.

We claim:

Proposition 2. *Let $T = (A, \lambda)$ be a tree automaton with multiplicities, and $T' = (A', \lambda')$ be the $\mathbb{N}$-VPA defined before. The set of accepting computations of T is in bijection with the set of accepting paths of T', and T' verifies property* (4).

[2] We use indexes to explicit whether the multiplicity is computed within T or within T'.

C.3 A New Algorithm for Finiteness of Weighted Tree Automata

We consider a trimmed weighted tree automaton $\mathcal{B} = (\Xi, Q, I, \Delta, \lambda)$ where Ξ is a ranked alphabet (with symbols of arity at most m, Ξ_i being symbols of arity i), Q is a finite set of states, $I \subseteq Q$ the set of initial states, Δ is a transition relation ($\Delta \subseteq \bigcup_{i=0}^{m} Q^i \times \Xi_i \times Q$) and λ associates with each transition rule a positive integer.

We construct by saturation three sets $\mathcal{S}_0$, $\mathcal{S}_1$ and $\mathcal{S}_2$ of tuples built from states and a Boolean respectively of the form (q_1, q_1', q_1'', b), $(q_1, q_1', q_2, q_2', q_3, q_3', b)$, (b) where $q_i, q_i' \in Q$ and $b \in \mathbb{B}$ using the following inference rules from Figure 13.

$$\frac{\tau = a \to q, \tau' = a \to q', \tau'' = a \to q'' \in \Delta}{(q, q', q'', \tau \neq \tau' \vee \lambda(\tau) > 1) \in \mathcal{S}_0} \text{ (1.1)}$$

$$\frac{\begin{array}{c}\text{the arity of } f \text{ is } k, \text{ for all } 1 \leq j \leq k, (q_j, q_j', q_j'', b_j) \in \mathcal{S}_0 \\ \tau = f(q_1, \ldots, q_k) \to q, \tau' = f(q_1', \ldots, q_k') \to q', \tau'' = f(q_1'', \ldots, q_k'') \to q'' \in \Delta\end{array}}{(q, q', q'', \bigvee_j b_j \vee \tau \neq \tau' \vee \lambda(\tau) > 1) \in \mathcal{S}_0} \text{ (1.2)}$$

$$\frac{p_i \in Q \text{ for all } i \in \{1, 2, 3\}}{(p_1, p_1, p_2, p_2, p_3, p_3, \bot) \in \mathcal{S}_1} \text{ (2.1)}$$

$$\frac{\begin{array}{c}\text{the arity of } f \text{ is } k, (p_j, q_j, p_j', q_j', p_j'', q_j'', b_j) \in \mathcal{S}_1 \text{ for some } 1 \leq j \leq k \\ \text{for all } 1 \leq \ell \leq k, \ell \neq j, (q_\ell, q_\ell', q_\ell'', b_\ell) \in \mathcal{S}_0 \\ \tau = f(q_1, \ldots, q_k) \to q, \tau' = f(q_1', \ldots, q_k') \to q', \tau'' = f(q_1'', \ldots, q_k'') \to q'' \in \Delta\end{array}}{(p_j, q, p_j', q', p_j'', q'', \bigvee_{j|j\neq\ell} b_j \vee \tau \neq \tau' \vee \lambda(\tau) > 1) \in \mathcal{S}_1} \text{ (2.2)}$$

$$\frac{(p, p, p, q, q, q, \top) \in \mathcal{S}_1}{(\top) \in \mathcal{S}_2} \text{ (3.1)}$$

Fig. 13. Inference rules for deciding finiteness.

Intuitively, a triple (q, q', q'', b) belongs to $\mathcal{S}_0$ if there exists a tree having three runs which label the root respectively by q, q' and q'' and b is true if the first one has used a transition whose multiplicity is strictly greater than 1 or the first two runs differ at some position in the transition rules they use. A tuple $(q_1, q_2, q_1', q_2', , q_1'', q_2'', b)$ belongs to $\mathcal{S}_1$ if there exists a context having three runs which label the hole of the context by q_1, q_1' and q_1'' respectively and the root by q_2, q_2' and q_2'' respectively and b is true if the first one has used a transition whose multiplicity is strictly greater than 1 or the first two runs differ at some position in the transition rules they use. Finally, the last rule is used to identify the presence of the required pattern.

Obviously, this algorithm is in PTIME.